Tesis – SM142501

MULTIPLE SEQUENCE ALIGNMENT MENGGUNAKAN *NATURE-INSPIRED METAHEURISTIC ALGORITHMS*

MUHAMMAD LUTHFI SHAHAB
1215 201 010

DOSEN PEMBIMBING
Prof. Dr. Mohammad Isa Irawan, M. T.

PROGRAM MAGISTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA

2017

Tesis – SM142501

MULTIPLE SEQUENCE ALIGNMENT MENGGUNAKAN *NATURE-INSPIRED METAHEURISTIC ALGORITHMS*

MUHAMMAD LUTHFI SHAHAB
1215 201 010

DOSEN PEMBIMBING
Prof. Dr. Mohammad Isa Irawan, M. T.

PROGRAM MAGISTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2017

Thesis – SM142501

MULTIPLE SEQUENCE ALIGNMENT USING NATURE-INSPIRED METAHEURISTIC ALGORITHMS

MUHAMMAD LUTHFI SHAHAB
1215 201 010

SUPERVISOR
Prof. Dr. Mohammad Isa Irawan, M. T.

MASTER DEGREE
MATHEMATICS DEPARTMENT
FACULTY OF MATHEMATICS AND NATURAL SCIENCES
SEPULUH NOPEMBER INSTITUTE OF TECHNOLOGY
SURABAYA
2017

MULTIPLE SEQUENCE ALIGNMENT MENGGUNAKAN *NATURE-INSPIRED METAHEURISTIC ALGORITHMS*

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Sains (M. Si.)
di
Institut Teknologi Sepuluh Nopember Surabaya

oleh :
MUHAMMAD LUTHFI SHAHAB
NRP. 1215 201 010

Tanggal Ujian : 9 Januari 2017
Periode Wisuda : Maret 2017

Disetujui oleh :

Prof. Dr. Mohammad Isa Irawan, M. T. (Pembimbing)
NIP. 19631225 198903 1 001

Dr. Imam Mukhlash, M. T. (Penguji)
NIP. 19700831 199403 1 003

Dr. Didik Khusnul Arif, M. Si. (Penguji)
NIP. 19730930 199702 1 001

Dr. Dieky Adzkiya, M. Si. (Penguji)
NIP. 19830517 200812 1 003

an. Direktur Program Pascasarjana
Asisten Direktur,

Prof. Dr. Ir. Tri Widjaja, M. Eng.
NIP. 19611021 198603 1 001

MULTIPLE SEQUENCE ALIGNMENT MENGGUNAKAN *NATURE-INSPIRED METAHEURISTIC ALGORITHMS*

Nama : Muhammad Luthfi Shahab
NRP : 1215 201 010
Pembimbing : Prof. Dr. Mohammad Isa Irawan, M. T.

ABSTRAK

Multiple sequence alignment adalah proses dasar yang sering dibutuhkan dalam mengolah beberapa *sequence* yang berhubungan dengan bioinformatika. Apabila *multiple sequence alignment* telah selesai dikerjakan, maka dapat dilakukan analisis-analisis lain yang lebih jauh, seperti analisis filogenetik atau prediksi struktur protein. Banyaknya kegunaan dari *multiple sequence alignment* mengakibatkannya menjadi salah satu permasalahan yang banyak diteliti. Banyak algoritma-algoritma *metaheuristic* yang berdasar pada kejadian-kejadian alami, yang biasa disebut dengan *nature-inspired metaheuristic algorithms*. Beberapa algoritma baru dalam *nature-inspired metaheuristic algorithms* yang dianggap cukup efisien antara lain adalah *firefly algorithm*, *cuckoo search*, dan *flower pollination algorithm*. Dalam penelitian ini dipaparkan *modified Needleman-Wunsch alignment*. Didapatkan hasil bahwa *modified Needleman-Wunsch alignment* adalah metode yang cukup bagus. *Modified Needleman-Wunsch alignment* tersebut digunakan untuk membentuk solusi awal dari *firefly algorithm*, *cuckoo search*, dan *flower pollination algorithm*. Didapatkan hasil bahwa *firefly algorithm*, *cuckoo search*, dan *flower pollination algorithm* dapat menghasilkan solusi-solusi baru yang lebih baik. Secara keseluruhan, *firefly algorithm* adalah algoritma yang terbaik dari tiga algoritma tersebut dalam segi skor *alignment*, namun membutuhkan waktu komputasi yang lebih besar.

Kata kunci: *multiple sequence alignment, modified Needleman-Wuncsh alignment, firefly algorithm, cuckoo search, flower pollination algorithm*

MULTIPLE SEQUENCE ALIGNMENT USING NATURE-INSPIRED METAHEURISTIC ALGORITHMS

Name : Muhammad Luthfi Shahab
NRP : 1215 201 010
Supervisor : Prof. Dr. Mohammad Isa Irawan, M. T.

ABSTRACT

Multiple sequence alignment is a fundamental tool that often needed to process bioinformatic sequences. If multiple sequence alignment is completed, we can process other further analysis, such as phylogenetic analysis or protein structure prediction. The versatility of multiple sequence alignment led it to be the one of the problems that studied continously. Many metaheuristic algorithms are based on natural events, with the so called nature-inspired metaheuristic algorithms. Algorithms in nature-inspired metaheuristic algorithms that considered to be good are firefly algorithm, cuckoo search, and flower pollination algorithm. In this research, we propose modified Needleman-Wunsch alignment. The results show that modified Needleman-Wunsch alignment is a good method. Modified Needleman-Wunsch alignment is used to create initial solution of firefly algorithm, cuckoo search, and flower pollination algorithm. The results show that firefly algorithm, cuckoo search, and flower pollination algorithm can produce new better solution. Overall, firefly algorithm is the best algorithm among the others in alignment score, but need large computation time.

Keywords: multiple sequence alignment, modified Needleman-Wuncsh alignment, firefly algorithm, cuckoo search, flower pollination algorithm

KATA PENGANTAR

Bismillahirrahmanirrahim.

Puji syukur penulis panjatkan ke hadirat Allah SWT, Maha Kuasa atas segala sesuatu, yang telah mengizinkan penulis untuk dapat menyelesaikan Tesis yang berjudul "*Multiple Sequence Alignment* Menggunakan *Nature-Inspired Metaheuristic Algorithms*". Tidak lupa, sholawat serta salam penulis haturkan kepada Rasulullah Muhammad SAW, atas cahaya lurus yang senantiasa Beliau sebarkan.

Ucapan terima kasih penulis sampaikan kepada pihak-pihak yang membantu dalam menyelesaikan Tesis ini, khususnya kepada:

1. Bapak Prof. Dr. Mohammad Isa Irawan, M. T. selaku dosen pembimbing atas segala bimbingan, saran, dukungan, kesabaran dan waktu yang diberikan kepada penulis hingga Tesis ini selesai.
2. Bapak Dr. Imam Mukhlash, M. T., Bapak Dr. Didik Khusnul Arif, M. Si., dan Bapak Dr. Dieky Adzkiya, M. Si. selaku dosen penguji atas kritik dan saran demi perbaikan Tesis ini.
3. Seluruh dosen Jurusan Matematika Institut Teknologi Sepuluh Nopember, atas ilmu yang telah diberikan selama penulis menempuh masa perkuliahan.
4. Walid tercinta, Zaid Hasan Shahab, dan Mama tercinta, Sri Rahayu, atas segenap cinta, kasih sayang, doa, serta perhatian yang tidak pernah henti diberikan untuk penulis.
5. Saudara-saudara tercinta, Mbak Evi, Mas Dafid, Mas Risqi, dan Nadiyya, dan seluruh keluarga yang selalu memberikan dukungan kepada penulis.
6. Cordova Ulin Nuha Kamila, S. Si. yang selalu memberikan saran, dukungan, dan semangat kepada penulis.
7. Teman-teman pejuang wisuda 115 yang telah berbagi suka dan duka dalam mengerjakan Tesis ini.
8. Semua pihak yang tidak dapat disebutkan satu-persatu yang telah membantu dalam pengerjaan Tesis ini sehingga dapat terselesaikan dengan baik.

Penulis menyadari bahwa Tesis ini masih jauh dari kesempurnaan baik dari segi teknik penulisan maupun materi, oleh karena itu kritik dan saran yang membangun sangat penulis harapkan demi kesempurnaan Tesis ini. Semoga Tesis ini dapat memberikan banyak manfaat bagi semua pihak.

Surabaya, Januari 2016

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB I PENDAHULUAN

1.1 Latar Belakang

Multiple sequence alignment adalah proses dasar yang sering dibutuhkan dalam mengolah beberapa *sequence* (umumnya lebih dari atau sama dengan tiga *sequence*) yang berhubungan dengan bioinformatika. Apabila *multiple sequence alignment* telah selesai dikerjakan, maka dapat dilakukan analisis-analisis lain yang lebih jauh. Misalnya adalah analisis filogenetik. Dengan analisis filogenetik, dapat dicari hubungan-hubungan evolusi yang terjadi pada berbagai *sequence* DNA suatu organisme. Peran dari *multiple sequence alignment* dalam analisis filogenetik adalah menyajikan estimasi yang akurat dari jarak-jarak setiap pasangan *sequence* tersebut. *Multiple sequence alignment* juga dapat digunakan untuk mengidentifikasi pola struktur dan fungsi dari beberapa *sequence* protein yang terkait. Selain itu, *multiple sequence alignment* juga dapat digunakan untuk memprediksi struktur protein. Prediksi struktur sekunder dan tersier bertujuan untuk memprediksi struktur *buried, exposed, helix, strand,* dan lain lain. Prediksi struktur sekunder yang berdasar pada satu *sequence* memiliki akurasi yang rendah, sekitar 60 persen, sedangkan prediksi yang berdasar pada *multiple sequence alignment* memiliki akurasi yang lebih tinggi, sekitar 75 persen (Notredame, 2002).

Saat ini banyak bermunculan algoritma-algoritma *metaheuristic* yang berdasar pada kejadian-kejadian alami, biasa disebut dengan *nature-inspired metaheuristic algorithms*. Terdapat banyak sekali *nature-inspired metaheuristic algorithms* yang dapat digunakan untuk menyelesaikan masalah optimasi. *Nature-inspired metaheuristic algorithms* yang lebih awal ditemukan antara lain adalah *genetic algorithm, ant algorithm, particle swarm optimization,* dan lain-lain.

Nature-inspired metaheuristic algorithms, terutama yang berdasar pada *swarm intelligence,* telah banyak dikembangkan dan dipelajari dalam sepuluh tahun terakhir. Misalnya *firefly algorithm* yang dikembangkan pada 2008 dan *cuckoo search* yang dikembangkan pada 2009. Algoritma lain yang juga baru

dikembangkan adalah *flower pollination algorithm* yaitu pada 2012. (Yang, 2014).

Firefly algorithm adalah algortima yang berdasar pada pola dan perilaku cahaya dari kawanan *firefly* (kunang-kunang). *Cuckoo search* adalah algoritma yang berdasar pada *brood parasitism* dari beberapa spesies *cuckoo* (burung kukuk). Dan *flower pollination algorithm* adalah algoritma yang berdasar pada proses penyerbukan bunga. Algoritma-algoritma tersebut memiliki banyak keunggulan dari algoritma-algoritma yang telah ditemukan sebelumnya. Dan untuk masalah optimasi fungsi, algoritma-algoritma tersebut biasanya dapat menghasilkan solusi yang lebih baik (Yang, 2014). Namun algoritma-algoritma tersebut belum digunakan untuk menyelesaikan *multiple sequence alignment*.

Berdasarkan hal tersebut, maka dalam penelitian ini akan dikaji penggunaan *firefly algorithm*, *cuckoo search*, dan *flower pollination algorithm* untuk menyelesaikan *multiple sequence alignment*. Perancangan setiap algorithma akan dilakukan secara runtut dan hasil dari setiap algoritma akan dibandingkan agar dapat diketahui algoritma yang terbaik dalam menyelesaikan *multiple sequence alignment*.

1.2 Rumusan Masalah

Rumusan masalah dalam penelitian ini adalah sebagai berikut:

1. Bagaimana perumusan *firefly algorithm* agar dapat digunakan untuk menyelesaikan *multiple sequence alignment*?
2. Bagaimana perumusan *cuckoo search* agar dapat digunakan untuk menyelesaikan *multiple sequence alignment*?
3. Bagaimana perumusan *flower pollination algorithm* agar dapat digunakan untuk menyelesaikan *multiple sequence alignment*?
4. Bagaimana perbandingan hasil yang diperoleh antara *firefly algorithm*, *cuckoo search*, dan *flower pollination algorithm* dalam menyelesaikan *multiple sequence alignment*?

1.3 Batasan Masalah

Batasan masalah yang digunakan dalam penelitian ini adalah sebagai berikut:

1. *Sequence* yang digunakan adalah *sequence* DNA.
2. Model gap yang digunakan adalah model gap linier.
3. Simulasi akan dilakukan dengan menggunakan bahasa pemrograman Java dalam NetBeans IDE 8.2.

1.4 Tujuan

Tujuan dari pengerjaan penelitian ini adalah sebagai berikut:

1. Merumuskan *firefly algorithm* agar dapat digunakan untuk menyelesaikan *multiple sequence alignment*.
2. Merumuskan *cuckoo search* agar dapat digunakan untuk menyelesaikan *multiple sequence alignment*.
3. Merumuskan *flower pollination algorithm* agar dapat digunakan untuk menyelesaikan *multiple sequence alignment*.
4. Membandingkan hasil yang diperoleh antara *firefly algorithm*, *cuckoo search*, dan *flower pollination algorithm* dalam menyelesaikan *multiple sequence alignment*.

1.5 Manfaat

Manfaat yang bisa diperoleh dari pengerjaan penelitian ini adalah sebagai berikut:

1. Adanya kajian baru tentang *firefly algorithm*, *cuckoo search*, dan *flower pollination algorithm* untuk menyelesaikan *multiple sequence alignment*. Nantinya, algoritma-algoritma tersebut dapat dikembangkan lebih lanjut oleh peneliti yang lain.
2. Mengetahui performansi dari *firefly algorithm*, *cuckoo search*, dan *flower pollination algorithm* dalam menyelesaikan *multiple sequence alignment*.
3. Menjadi sarana bagi penulis untuk bisa lebih memahami bidang keilmuan *nature-inspired metaheuristic algorithms*.

BAB II TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Algoritma-algoritma yang biasa digunakan untuk menyelesaikan *multiple sequence alignment* dapat diklasifikasikan dalam tiga metode utama yaitu metode eksak, metode progresif, dan metode iteratif (Naznin dkk, 2011).

Metode eksak sangat baik apabila digunakan untuk menemukan *alignment* dari dua *sequence* (Needleman dan Wunsch, 1970). Namun, kompleksitas dari metode tersebut berkembang sangat pesat apabila digunakan pada tiga atau lebih *sequence* (Stoye dkk, 1997).

Multiple sequence alignment dengan metode progresif (metode yang berdasar pada *tree*) diantaranya adalah MULTALIGN (Barton dan Sternberg, 1987), MULTAL (Taylor, 1988), PILEUP (Devereux dkk, 1984), CLUSTALX (Thompson dkk, 1997), CLUSTAL W (Thompson dkk, 1994), dan T-Coffee (Notredame dkk, 1987). MULTAL bekerja dengan menjajarkan *sequence-sequence* yang mirip. MULTALIGN, PILEUP, dan CLUSTAL W bekerja dengan menggunakan *guide tree* yang dibentuk dengan suatu cara tertentu. Sedangkan T-Coffee bekerja dengan mengkombinasikan informasi dari *alignment* lokal dan global. Kerugian dari metode progresif adalah kemungkinan terjebaknya pada optimum lokal (Thompson dkk, 1994). Untuk menangani masalah tersebut maka mulai digunakan metode iteratif.

Metode iteratif banyak yang berdasar pada *evolutionary algorithms*. *Evolutionary algorithms* adalah algoritma stokastik yang menggunakan prinsip populasi. *Evolutionary algorithms* menggunakan representasi solusi yang berbeda-beda dan menggunakan beberapa operator reproduksi untuk menghasilkan solusi baru. Algoritma yang paling sering digunakan adalah *genetic algorithm*. Solusi awal biasanya dibuat dengan menggunakan metode progresif (Naznin dkk, 2011). Sebagai contoh MSA-EA (Thomsen dkk, 2002) meningkatkan hasil dari CLUSTAL V (Higgins dkk, 1992). Beberapa yang berdasar pada *genetic algorithm* antara lain SAGA (Notredame dan Higgins, 1996), MSA-EC (Shyu dkk, 2004), dan GA-ACO (Lee dkk, 2008).

2.2 *Sequence* DNA

Sequence DNA, *deoxyribonucleic acid*, (selanjutnya hanya akan disebut dengan *sequence*) adalah barisan dari empat macam *nucleotide*, yaitu *adenine* (A), *cytosine* (C), *guanine* (G), dan *thymine* (T) (Isaev, 2006). Sebagai contoh, TGTTCTCGATAGTACTTGCA adalah *sequence* yang terdiri dari dua puluh *nucleotide*.

Dari *sequence* yang sudah ada, dapat terbentuk *sequence* baru apabila terjadi suatu mutasi pada *nucleotide-nucleotide* dari *sequence* tersebut. Secara umum, terdapat empat tipe mutasi yang bisa terjadi yaitu (Shen dan Tuszynski, 2008):

1. Mutasi tipe 1, adalah mutasi yang disebabkan suatu *nucleotide* berubah menjadi *nucleotide* lainnya, misalnya A pada TGTTCAGTA berubah menjadi G sehingga didapat TGTTCGGTA.
2. Mutasi tipe 2, adalah mutasi yang disebabkan suatu *nucleotide* bertukar posisi dengan *nucleotide* didepan atau dibelakangnya, misalnya A bertukar posisi dengan C pada *sequence* TGTTCAGTA sehingga didapat TGTTACGTA.
3. Mutasi tipe 3, adalah mutasi yang disebabkan adanya suatu *nucleotide* yang disisipkan pada *sequence*, misalnya G disisipkan pada TGTTCAGTA sehingga didapat TGTGTCAGTA.
4. Mutasi tipe 4, adalah mutasi yang disebabkan adanya suatu *nucleotide* pada *sequence* yang dihapus, misalnya T pada TGTTCAGTA dihapus sehingga didapat TGTCAGTA.

Selanjutnya, sembarang *sequence* dapat disimbolkan dengan s_i untuk suatu $i \in \mathbb{N}$ dan *nucleotide* dari *sequence* adalah elemen dari $N_4 = \{\text{A},\text{C},\text{G},\text{T}\}$.

2.3 *Multiple Sequence Alignment*

Secara umum, *multiple sequence alignment* adalah proses penjajaran n *sequence*, dimana $n \geq 3$. Perhatikan penjelasan secara matematis mengenai *multiple sequence alignment* berikut ini.

Misalkan terdapat n *sequence*, yaitu $s_1, s_2, \dots, s_n$, yang tersusun dari elemen-elemen dari N_4. Kemudian disisipkan beberapa gap "−" ke dalam

$s_1, s_2, \dots, s_n$ sehingga didapatkan n *sequence*, yaitu $s_1', s_2', \dots, s_n'$, yang tersusun dari elemen-elemen dari $N_5 = \{\mathtt{A},\mathtt{C},\mathtt{G},\mathtt{T},-\}$.

Definisi 2.1 (Shen dan Tuszynski, 2008)

1. *Sequence* s_1' disebut ekspansi dari s_1 apabila s_1' diperoleh dengan menyisipkan beberapa gap "$-$" ke dalam s_1.
2. *Sequence* s_1', s_2' disebut ekspansi dari s_1, s_2 apabila s_1', s_2' berturut-turut adalah ekspansi dari s_1, s_2.
3. *Multiple sequence* $\mathcal{S}' = \{s_1', s_2', \dots, s_n'\}$ disebut ekspansi dari *multiple sequence* $\mathcal{S} = \{s_1, s_2, \dots, s_n\}$ apabila setiap s_i' adalah ekspansi dari s_i.
4. $\mathcal{S}$ disebut *sequence* asal dari $\mathcal{S}'$ apabila *multiple sequence* $\mathcal{S}'$ adalah ekspansi dari $\mathcal{S}$. Kemudian dinotasikan *sequence* dalam $\mathcal{S}'$ dengan

$$s_i' = s_{i,1}' s_{i,2}' \dots s_{i,l_i}' \tag{2.1}$$

   dimana $s_{i,j}' \in N_5$.
5. *Multiple sequence* $\mathcal{S}'$ disebut *multiple sequence alignment* dari $\mathcal{S}$ jika $l_1 = l_2 = \cdots = l_n = L$, jika $\mathcal{S}'$ adalah ekspansi dari $\mathcal{S}$, dan jika terdapat salah satu dari $s_{1,j}', s_{2,j}', \dots, s_{n,j}'$ yang bukan merupakan gap "$-$" untuk $1 \leq j \leq L$.

2.3.1 Skor dari *Multiple Sequence Alignment*

Tujuan dari *multiple sequence alignment* adalah untuk mencari ekspansi $\mathcal{S}'$ dari *sequence-sequence* dalam $\mathcal{S}$ sedemikian sehingga $\mathcal{S}'$ memiliki skor yang besar. Skor yang besar menandakan bahwa $\mathcal{S}'$ adalah hasil penjajaran yang banyak menempatkan *nucleotide-nucleotide* yang sama pada letak yang benar. Skor tersebut biasa dihitung dengan memanfaatkan suatu matriks skor. Secara umum matriks skor tersebut, misalkan M, ditampilkan dalam Tabel 2.1 (Shen dan Tuszynski, 2008).

Tabel 2.1 Matriks Skor *M*

M	A	C	G	T	–
A	1	0	0	0	-2
C	0	1	0	0	-2
G	0	0	1	0	-2
T	0	0	0	1	-2
–	-2	-2	-2	-2	0

Keseluruhan skor dari $\mathcal{S}'$ dihitung dengan

$$f(\mathcal{S}') = \sum_{j=1}^{L} \sum_{a<b} M\left(s'_{a,j}, s'_{b,j}\right). \tag{2.2}$$

Semakin besar nilai dari $f(\mathcal{S}')$, maka semakin baik hasil dari *multiple sequence alignment* yang diperoleh.

2.3.2 *Needleman-Wunsch Alignment*

Needleman-Wunsch alignment adalah metode untuk mendapatkan seluruh *alignment* global yang optimal, biasanya terdapat lebih dari satu *alignment* global yang optimal. Misalkan terdapat dua *sequence*, $s_1 = a_1 a_2 \dots a_i \dots a_n$ dan $s_2 = b_1 b_2 \dots b_j \dots b_m$. Dibentuk sebuah matriks F dengan ordo $(n+1) \times (m+1)$. Elemen pada baris ke-i dan kolom ke-j, $F(i,j)$, untuk $1 \leq i \leq n$ dan $1 \leq j \leq m$ adalah sama dengan skor *alignment* optimal antara $a_1 a_2 \dots a_i$ dan $b_1 b_2 \dots b_j$. Elemen $F(i,0)$, untuk $1 \leq i \leq n$ adalah skor dari *alignment* antara $a_1 a_2 \dots a_i$ dan gap sepanjang i. Begitu pula Elemen $F(0,j)$, untuk $1 \leq j \leq m$ adalah skor dari *alignment* antara $b_1 b_2 \dots b_j$ dan gap sepanjang j. Matriks F dibentuk secara rekursif dengan memberikan inisialisasi $F(0,0) = 0$ dan kemudian mengisi setiap elemen matriks mulai dari sisi kiri atas menuju sisi kanan bawah. Apabila $F(i-1, j-1)$, $F(i-1, j)$, dan $F(i, j-1)$ diketahui, $F(i,j)$ dihitung dengan menggunakan

$$F(i,j) = \max \begin{cases} F(i-1, j-1) + M\left(a_i, b_j\right), \\ F(i-1, j) + M(a_i, -), \\ F(i, j-1) + M\left(-, b_j\right). \end{cases} \tag{2.3}$$

Saat menghitung $F(i,j)$, arah yang menuju diperolehnya $F(i,j)$ disimpan. Saat telah diperoleh $F(n,m)$ dilakukan *back track* untuk mendapatkan *alignment* optimal. Nilai dari $F(n,m)$ adalah skor dari *alignment* yang didapat (Isaev, 2006).

Perhatikan contoh berikut untuk lebih memahami cara menggunakan *Needleman-Wunsch alignment*. Misalkan $s_1 = \texttt{CTTAGA}$ dan $s_2 = \texttt{GTAA}$. Matriks F untuk *sequence* tersebut adalah seperti yang ditampilkan dalam Tabel 2.2.

Tabel 2.2 Matriks Skor *F*

F	–	G	T	A	A
–	0	-2	-4	-6	-8
C	-2	0	-2	-4	-6
T	-4	-2	1	-1	-3
T	-6	-4	-1	1	-1
A	-8	-6	-3	0	2
G	-10	-7	-5	-2	0
A	-12	-9	-7	-4	-1

Apabila dilakukan *back track* akan didapatkan tiga *alignment* yaitu

```
C T T A G A
G – T A – A

C T T A G A
– G T A – A

C T T A G A
G T – A – A
```

yang semuanya memiliki skor *alignment* -1.

Untuk menggunakan *Needleman-Wunsch alignment* untuk lebih dari tiga *sequence*, maka perlu dilakukan pengembangan lagi dari (2.4). *Pseudo code* dari *Needleman-Wunsch alignment* untuk dua *sequence* ditampilkan dalam Gambar 2.1 dan *pseudo code* untuk *back track* ditampilkan dalam Gambar 2.2. *Back track* tersebut hanya akan mengambil satu *sequence* global optimal agar waktu komputasinya tidak terlalu besar.

```
needleman_wunsch_alignment
    input  : a
             b
    output : s

    F(0,0) = 0
    for i = 1 to length(a)+1
        F(i,0) = F(i-1,0) + score(a(i),-)
    for i = 1 to length(a)+1
        F(0,i) = F(0,i-1) + score(b(i),-)
    for i = 1 to length(a)+1
        for j = 1 to length(b)+1
            f(1) = F(i-1,j-1) + score(a(i),b(j))
            f(2) = F(i,j-1) + score(-,b(j))
            f(3) = F(i-1,j) + score(a(i),-)
            F(i,j) = max(f)
    s = back_track(F, a, b)
    return s
```

Gambar 2.1 *Pseudo Code* untuk *Needleman-Wunsch Alignment*

Apabila terdapat tiga *sequence* yaitu $s_1 = a_1 a_2 \dots a_i \dots a_{n_1}$, $s_2 = b_1 b_2 \dots b_j \dots b_{n_2}$, dan $s_3 = c_1 c_2 \dots c_k \dots c_{n_3}$ maka $F(i,j,k)$ dihitung dengan

$$F(i,j,k) = \max \begin{cases} F(i-1,j-1,k-1) + M(a_i, b_j, c_k), \\ F(i,j-1,k-1) + M(-, b_j, c_k), \\ F(i-1,j,k-1) + M(a_i, -, c_k), \\ F(i-1,j-1,k) + M(a_i, b_j, -), \\ F(i,j,k-1) + M(-,-,c_k), \\ F(i,j-1,k) + M(-, b_j, -), \\ F(i-1,j,k) + M(a_i, -, -). \end{cases} \tag{2.4}$$

2.3.3 *Star Alignment*

Star alignment adalah algoritma untuk menyelesaikan *multiple sequence alignment* yang memiliki waktu komputasi sangat cepat. Namun *star alignment* tidak menjamin akan ditemukannya *alignment* yang optimal. Ide dasar dari *star alignment* adalah dengan pertama-tama menemukan satu *sequence* yang paling mirip dengan semua *sequence* yang lain dan kemudian menggunakannya sebagai *star* dan menjajarkan semua *sequence* dengannya (Isaev, 2006).

```
back_track
    input  : F
             a
             b
    output : s

    c = ""
    i = length(a)
    j = length(b)

    while i > 0 or j > 0
        if j == 0
            c = a(i-1) + "-" + c
            i--
        else if i == 0
            c = "-" + b (j-1) + c
            j--
        else
            f(1) = F(i-1,j-1) + score(a(i),b(j))
            f(2) = F(i,j-1) + score(-,b(j))
            f(3) = F(i-1,j) + score(a(i),-)
            if F(i,j) == f(1)
                c = a(i-1) + "" + b(j-1) + c
                i--
                j--
            else if F(i,j) == f(3)
                c = a(i-1) + "-" + c
                i--
            else if F(i,j) == f(2)
                c = "-" + b(j-1) + c
                j--
    d = ""
    e = ""
    for i = 0 to length(c)/2
        d = d + c(2*i)
        e = e + c(2*i+1)
    s[] = {d, e};
    return s;
```

Gambar 2.2 *Pseudo Code* untuk *Back Track*

Perhatikan contoh berikut untuk lebih memahami cara menggunakan *star alignment*. Misalkan $s_1 =$ GGCAA, $s_2 =$ GCACA, dan $s_3 =$ GGCA. Dengan menggunakan *Needleman-Wunsch alignment* untuk dua *sequence* didapatkan

$$s_1 = \texttt{G G C A A}$$

$$s_2 = \texttt{G C A C A}$$

dengan skor 2,

$s_1 =$ G G C A A

$s_3 =$ G G C - A

dengan skor 2, dan

$s_2 =$ G C A C A

$s_3 =$ G G - C A

dengan skor 1. Sehingga skor untuk setiap pasangan adalah seperti yang ditunjukkan dalam Tabel 2.3.

Tabel 2.3 Skor untuk Setiap Pasangan *Sequence*

	s_1	s_2	s_3	Jumlah
s_1	-	2	2	4
s_2	2	-	1	3
s_3	2	1	-	3

Karena jumlah skor terbesar adalah milik s_1, maka yang digunakan sebagai *star* adalah s_1. Sehingga hasil *alignment* antara s_1, s_2, dan s_3 adalah

G G C A A

G C A C A

G G C - A

yang memiliki skor 4.

Pseudo code untuk *star alignment* ditampilkan dalam Gambar 2.3 dan *pseudo code* untuk *combine* yang digunakan dalam *star alignment* ditampilkan dalam Gambar 2.4.

2.4 *Firefly Algorithm*

Cahaya kunang-kunang adalah pemandangan yang sangat menarik. Terdapat sekitar dua ribu spesies kunang-kunang, dan kebanyakan dari mereka mengeluarkan cahaya singkat yang mempunyai ritme tertentu. Pola dari ritme tersebut biasanya berbeda-beda, tergantung dari spesiesnya. Ritme cahaya tersebut mempunyai dua fungsi utama, yaitu untuk menarik perhatian kunang-kunang lain dan untuk menarik perhatian mangsa. Intensitas cahaya yang berjarak r dari pusat cahayanya mengikuti hukum *inverse-square*, yaitu intensitas cahaya I mengecil

seiring bertambahnya r. Intensitas cahaya juga mengecil karena terserap oleh udara. Hal ini menyebabkan kunang-kunang dapat melihat kawannya dalam jarak maksimal tertentu.

Firefly algorithm bekerja dengan memperhatikan aturan-aturan berikut ini:

- Semua kunang-kunang adalah *unisex*, yaitu setiap kunang-kunang akan tertarik dengan kunang-kunang lain tanpa memperhatikan jenis kelaminnya.
- Besarnya ketertarikan sebanding dengan intensitas cahaya yang dikeluarkan oleh kunang-kunang. Kunang-kunang dengan cahaya yang lebih redup akan bergerak kepada yang lebih terang. Intensitas cahaya suatu kunang-kunang akan berkurang seiring dengan bertambahnya jarak. Apabila tidak ada kunang-kunang yang lebih terang dari yang lain, maka kunang-kunang akan bergerak secara acak.
- Intensitas cahaya dari kunang-kunang ditentukan dari fungsi objektif.

```
star_alignment
    input  : a (array string[])
             n (ukuran dari a)
    output : s

    score = needleman_wunsch_pairs_score(a)
    for i = 1 to n
        row_score(i) = 0
        for j = 1 to n
            row_score(i) = row_score(i) + score(i)(j)
    star = 0
    max = row_score(1);
    for i = 2 to n
        if row_score(i) > max
            max = row_score(i);
            star = i
    if star ~= 1
        b = a(star);
        a(star) = a(1);
        a(1) = b;

    s = needleman_wunsch_alignment(a(1), a(2));
    for i = 3 to n
        S = needleman_wunsch_alignment(a[1], a[i]);
        s = combine(S, s);
    }
    return s
```

Gambar 2.3 *Pseudo Code* untuk *Star Alignment*

```
combine
    input  : s1 (array string[])
             s2 (array string[])
    output : s

    i = 1
    j = 1
    n1 = s1.length;
    for k = 1 to n1+1
        s[k] = ""

    while i < s1[1].length() and j < s2[1].length()
        if s1[1].charAt(i) == s2[1].charAt(j)
            for k = 1 to n1
                s[1] = s[1] + s1[1].charAt(i)
            s[n1] = s[n1] + s2[2].charAt(j)
            i = i + 1
            j = j + 1

        else if s1[1].charAt(i) == '-'
            for k = 1 to n1
                s[k] = s[k] + s1[k].charAt(i)
            s[n1] = s[n1] + "-"
            i = i + 1

        else if s2[1].charAt(j) == '-'
            for k = 1 to n1
                s[k] = s[k] + "-"
            s[n1] = s[n1] + s2[2].charAt(j)
            j = j + 1

    while i < s1[1].length()
        for k = 1 to n1
            s[k] = s[k] + s1[k].charAt(i)
        s[n1] = s[n1] + "-"
        i = i + 1

    while j < s2[1].length()
        for int k = 1 to n1
            s[k] = s[k] + "-"
        s[n1] = s[n1] + s2[2].charAt(j)
        j = j + 1

    return s;
```

Gambar 2.4 *Pseudo Code* untuk *Combine*

Untuk sebuah masalah optimasi, intensitas cahaya kunang-kunang bisa dibuat serupa dengan fungsi objektif. Berdasarkan tiga aturan tersebut, langkah-langkah *firefly algorithm* dapat diringkas seperti yang ditunjukkan dalam *pseudo code* dalam Gambar 2.5. Dalam *pseudo code* tersebut, solusi terbaik dari beberapa solusi dicari dengan pertama-tama mengurutkan semua solusi (*quick sort*) berdasarkan *fitness*-nya dan kemudian mengambil solusi pada urutan pertama.

```
firefly_algorithm
    input  : n (banyak solusi)
             x (solusi berukuran n)
             toleransi
             max_iterasi
    output : x* (solusi terbaik)

    iterasi = 0
    x* = best(x)
    while iterasi < max_iterasi and f(x*) > toleransi
        for i = 1 to n
            for j = 1 to n
                if f(x(j)) > f(x(i))
                    hitung x_baru(i) dengan (2.8)
                    x(i) = x_baru(i)
        x* = best(x)
        iterasi = iterasi + 1
    return x*
```

Gambar 2.5 *Pseudo Code* untuk *Firefly Algorithm*

Intensitas cahaya yang berjarak r dari pusat cahayanya $I(r)$ mengikuti hukum *inverse-square* yaitu

$$I(r) = \frac{I_0}{r^2} \tag{2.5}$$

dimana I_0 adalah intensitas cahaya sumber. Intensitas cahaya juga terserap oleh cahaya. Jika digunakan koefisien penyerapan cahaya γ, maka

$$I(r) = I_0 e^{-\gamma r} \tag{2.6}$$

Untuk menghindari singularitas pada persamaan (2.5) dan menggabungkannya dengan persamaan (2.6), maka digunakan

$$I(r) = I_0 e^{-\gamma r^2} \tag{2.7}$$

Pergerakan dari kunang-kunang i ke arah kunang-kunang j diberikan oleh

$$x_i^{t+1} = x_i^t + I_0 e^{-\gamma r_{ij}^2}\left(x_j^t - x_i^t\right) + \alpha \epsilon_i^t \quad (2.8)$$

dimana suku kedua pada sisi kanan persamaan tersebut menyatakan pergerakan ke arah kunang-kunang j dan suku ketiga menyatakan pergerakan acak dengan α adalah parameter, dan ϵ_i adalah vektor bilangan acak yang diambil dari distribusi normal atau distribusi uniform. r_{ij} adalah jarak antara kunang-kunang i dengan kunang-kunang j (Yang, 2014).

Firefly algorithm telah dikembangkan dan digunakan untuk menyelesaikan banyak permasalahan seperti *travelling salesman problem* (Jati dan Suyanto, 2011), *unit commitment problem for a power system* (Rampriya dkk, 2010), *combinatorial graph-coloring problem* (Fister dkk, 2012), *economic emission load dispatch problem* (Apostolopoulus, 2011), *multi-objective optimization* (Yang, 2013), dan lain-lain.

2.5 *Cuckoo Search*

Cuckoo adalah burung yang menarik, bukan hanya karena suaranya yang indah yang mereka buat namun juga karena strategi reproduksi mereka yang sangat cepat. Beberapa spesies meletakkan telurnya pada sarang umum. Mereka juga bisa membuang telur dari burung lain untuk meningkatkan kemungkinan menetasnya telur mereka. Beberapa spesies yang lain melakukan *brood parasitism*, yaitu meletakkan telurnya pada sarang dari burung *host* lain (biasanya spesies burung yang berbeda dengan burung *cuckoo*).

Burung *cuckoo* bisa saja berkelahi dengan burung *host*. Apabila burung *host* menemukan telur yang bukan miliknya, mereka bisa membuang telur tersebut atau justru pergi meninggalkan sarang tersebut dan membangun sarang yang baru. Terdapat beberapa spesies *cuckoo* yang dapat meniru warna dan pola dari beberapa telur yang ada dalam sarang burung *host*. Hal ini akan menurunkan kemungkinan dibuangnya telur mereka oleh burung *host*.

Selain itu, burung cuckoo dapat meletakkan telurnya pada waktu yang baik. Mereka akan memilih sarang dimana burung *host* baru saja menelur. Secara umum, telur dari burung *cuckoo* menetas lebih awal dari telur burung yang lain.

Bayi burung *cuckoo* tersebut akan menyingkirkan telur-telur yang lain dari sarang dan mengambil alih makanan-makanan yang diberikan oleh burung *host*.

Cuckoo search bekerja dengan memperhatikan aturan-aturan berikut ini:

- Setiap cuckoo menghasilkan satu telur untuk satuan waktu tertentu dan meletakkannya pada suatu sarang yang dipilih secara acak.
- Sarang terbaik dengan telur yang berkualitas tinggi akan dibawa ke iterasi selanjutnya.
- Banyaknya sarang *host* yang tersedia adalah tetap dan telur yang ditinggalkan oleh burung cuckoo dapat ditemukan oleh burung host dengan probabilitas $p_a \in (0, 1)$. Burung *host* dapat membuang telur tersebut atau justru meninggalkan sarangnya dan membuat sarang yang baru.

Cuckoo search menggunakan kombinasi yang seimbang antara pergerakan lokal dan pergerakan global yang diatur oleh parameter p_a. Pergerakan lokal diformulasikan dengan

$$x_i^{t+1} = x_i^t + s\left(x_j^t - x_k^t\right) \tag{2.9}$$

dimana x_j^t dan x_k^t adalah dua solusi yang berbeda yang dipilih secara acak dengan permutasi random dan s adalah *step size*.

Di sisi lain, pergerakan global dilakukan dengan *Levy flight*

$$x_i^{t+1} = x_i^t + \alpha L(s, \lambda) \tag{2.10}$$

dimana α adalah faktor skala (Yang, 2014).

Cuckoo search telah dikembangkan dan digunakan untuk menyelesaikan banyak permasalahan seperti *knapsack problem* (Layeb, 2011), *multi-objective scheduling problem* (Chandrasekaran dan Simon, 2012), *training feedforward neural networks* (Valian dkk, 2011), dan lain-lain.

Langkah-langkah *firefly algorithm* dapat diringkas seperti yang ditunjukkan dalam *pseudo code* dalam Gambar 2.6.

```
cuckoo_search
    input  : n (banyak solusi)
             x (solusi berukuran n)
             toleransi
             max_iterasi
             p_a
    output : x* (solusi terbaik)

    iterasi = 0
    x* = best(x)
    while iterasi < max_iterasi and f(x*) > toleransi
        for i = 1 to n
            hitung x_baru(i) dengan (2.10)
        if f(x_baru(i)) > f(x(i))
            x(i) = x_baru(i)
        for i = 1 to n
            r = random(0,1)
            if r < p_a
                hitung x_baru(i) (2.9)
                x(i) = x_baru(i)
        x* = best(x)
        iterasi = iterasi + 1
    return x*
```

Gambar 2.6 *Pseudo Code* untuk *Cuckoo Search*

2.6 *Flower Pollination Algorithm*

Penyerbukan bunga adalah proses perpindahan serbuk sari yang biasanya dibantu oleh serangga atau hewan lainnya. Terdapat beberapa bunga dan serangga tertentu yang membuat suatu hubungan yang biasa disebut dengan *flower-pollinator partnership*. Bunga-bunga tersebut hanya akan menarik serangga yang terlibat dalam hubungan tersebut, dan serangga itulah yang dianggap sebagai penyerbuk utama bagi bunga-bunga tersebut.

Penyerbukan dapat dibedakan menjadi dua tipe, penyerbukan abiotik dan penyerbukan biotik. Sekitar 90 persen dari tanaman berbunga menggunakan penyerbukan biotik, yaitu penyerbukan yang dibantu oleh serangga. Sisanya menggunakan penyerbukan abiotik yang biasa dibantu oleh angin. Beberapa serangga biasa mengunjungi bunga-bunga tertentu dan meninggalkan bunga-bunga yang lain. Fenomena tersebut biasa disebut *flower constancy*. Semua bunga yang mempunyai sifat *flower constancy* memiliki jaminan akan bereproduksi secara maksimal.

Penyerbukan bunga terjadi melalui penyerbukan silang atau penyerbukan sendiri. Dalam penyerbukan silang, serbuk sari akan jatuh ke bunga lain. Penyerbukan silang dan biotik terjadi pada jarak yang jauh melalui bantuan serangga. Penyerbukan ini biasa disebut dengan penyerbukan global. Pergerakan serangga akan dianggap sebagai pergerakan diskrit yang mengikuti distribusi Levy. Dalam penyerbukan sendiri, serbuk sari akan jatuh ke bunga yang sama atau yang sejenis. Penyerbukan sendiri biasanya tidak membutuhkan bantuan serangga.

Karakteristik dari penyerbukan, *flower constancy*, dan kebiasaan serangga dapat dijadikan dasar untuk membuat aturan-aturan berikut ini:

1. Penyerbukan silang dan biotik dianggap sebagai penyerbukan global dan serangga bergerak mengikuti distribusi Levy.
2. Penyerbukan sendiri dan abiotik dianggap sebagai penyerbukan lokal.
3. *Flower constancy* dapat dianggap sebagai rasio reproduksi yang sebanding dengan kemiripan antara dua bunga.
4. Penyerbukan lokal terjadi dengan probabilitas yang lebih tinggi dari penyerbukan global. Penyerbukan tersebut akan dikontrol dengan suatu nilai $p \in (0,1)$.

Dalam penyerbukan global, serbuk sari bisa jatuh pada bunga terbaik. Jika dimisalkan bunga terbaik adalah g_*, maka *flower constancy* dan aturan pertama dapat diformulasikan secara matematis dengan

$$x_i^{t+1} = x_i^t + \gamma L(g_* - x_i^t) \tag{2.11}$$

dimana x_i^t adalah serbuk sari i atau solusi x_i pada generasi t, g_* adalah solusi pada generasi t, γ adalah faktor skala, dan L adalah kekuatan penyerbukan yang diambil dari distribusi Levy.

Penyerbukan lokal dan *flower constancy* dapat diformulasikan secara matematis dengan

$$x_i^{t+1} = x_i^t + \varepsilon\left(x_j^t - x_k^t\right) \tag{2.12}$$

dimana x_j^t dan x_k^t adalah serbuk sari yang datang dari bunga lain yang masih sejenis. Hal ini mensimulasikan *flower constancy* pada ketetanggaan kecil. Nilai ε diambil dari distribusi uniform pada $(0, 1)$ (Yang, 2014 dan Nabil, 2016).

Flower pollination algorithm telah dikembangkan dan digunakan untuk menyelesaikan permasalahan-permasalahan seperti *antenna positioning problem* (Dahi dkk, 2016), *economic* dan *emission dispatch* (Abdelaziz dkk, 2016), dan *cluster analysis* (Wang dkk, 2016), dan lain-lain.

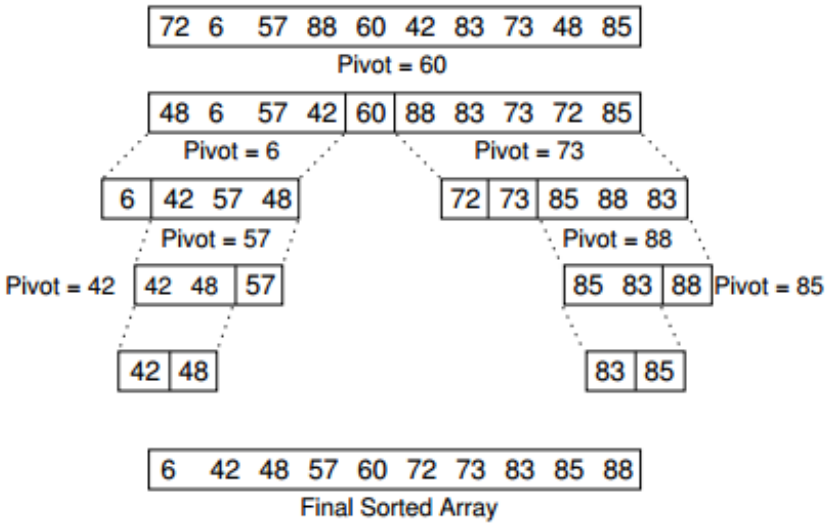
```
flower_pollination_algorithm
    input  : n (banyak solusi)
             x (solusi berukuran n)
             toleransi
             max_iterasi
             p
    output : x* (solusi terbaik)

    iterasi = 0
    x* = best(x)
    while iterasi < max_iterasi and f(x*) > toleransi
        for i = 1 to n
            r = random(0,1)
            if r < p
                hitung x_baru(i) dengan (2.11)
            else
                hitung x_baru(i) dengan (2.12)
        if f(x_baru(i)) > f(x(i))
            x(i) = x_baru(i)
        x* = best(x)
        iterasi = iterasi + 1
    return x*
```

Gambar 2.7 *Pseudo Code* untuk *Flower Pollination Algorithm*

2.7 *Quick Sort*

Diperlukan suatu mekanisme *sorting* yang dapat digunakan untuk mengurutkan solusi dengan *fitness* yang terbaik sampai yang terburuk. Terdapat banyak mekanisme umum yang dapat digunakan untuk *sorting*. *Quick sort* adalah salah satu mekanisme *sorting* yang cukup baik karena termasuk dalam $\Theta(n \log n)$. Ilustrasi berjalannya *quick sort* ditampilkan dalam Gambar 2.8, *pseudo code*-nya ditampilkan dalam Gambar 2.9, dan *pseudo code* dari *partition* ditampilkan dalam Gambar 2.10 (Shaffer, 2013).



Gambar 2.8 Ilustrasi dari *Quick Sort*

```
quick_sort
    input  : A (array yang belum disorting)
             i
             j
    output : A (array yang sudah disorting)

    pivot = (i+j)/2
    tukar A(j) dengan A(pivot)
    k = partition(A, i-1, j, A(j))
    tukar A(k) dengan A(j)
    if k-i > 1
        quick_sort(A, i, k-1)
    if j-k > 1
        quick_sort(A, k+1, j)
    return A
```

Gambar 2.9 *Pseudo Code* untuk *Quick Sort*

```
partition
    input  : A (array yang belum disorting)
             l
             r
             pivot
    output : l

    while (l < r)
        while A(++l) < pivot
        while r != 0 and A(--r) > pivot
        tukar A(l) dengan A(r)
    tukar A(l) dengan A(r)
    return l
```

Gambar 2.10 *Pseudo Code* untuk *Partition*

2.8 *Elitism Replacement with Filtration*

Elitism replacement digunakan agar solusi-solusi dalam populasi terus berkembang menjadi lebih baik. Misalkan populasi awal terdiri dari n solusi. *Elitism replacement* berjalanan dengan menggabungkan n solusi pada populasi awal dengan n solusi pada populasi baru menjadi satu populasi besar, sehingga terdiri dari $2n$ solusi. Kemudian solusi-solusi pada populasi besar tersebut diurutkan mulai dari yang terbaik hingga yang terburuk. Dilakukan filtrasi terhadap populasi besar tersebut, yaitu menghapus solusi apabila ia identik dengan solusi yang lainnya. n solusi terbaik yang didapatkan setelah dilakukan filtrasi akan diambil dan disimpan dalam populasi baru yang sesungguhnya. *Pseudo code* dari *elitism replacement with filtration* ditampilkan dalam Gambar 2.11 (Shahab dkk, 2016).

```
elitism_replacement_with_filtration
    input  : P1 (populasi awal yang terdiri dari n solusi)
             P2 (populasi baru yang terdiri dari n solusi)
    output : P (populasi akhir yang terdiri dari n solusi)

    bentuk P3 (populasi baru)
    for i = 1 to length(P1)
        P3(i) = P1(i)
        P3(length(P1)+i) = P2(i)
    quick_sort(P3)
    m = 1
    P(1) = P3(1)
    i = 2
    while i <= length(P3) and m < length(P1)
        if f(P3(i)) ~= f(P(m))
            m = m + 1
            P(m) = P3(i)
        i = i + 1
    return P
```

Gambar 2.11 *Pseudo Code* untuk *Elitism Replacement with Filtration*

BAB III METODE PENELITIAN

3.1 Studi Literatur

Dilakukan studi literatur untuk mendukung pengerjaan penelitian ini dan pemahaman yang lebih mendalam mengenai *multiple sequence alignment*, *firefly algorithm*, *cuckoo search*, dan *flower pollination algorithm*. Literatur yang dipelajari dapat bersumber dari jurnal, buku, internet, maupun bimbingan dengan dosen pembimbing.

3.2 Perumusan *Firefly Algorithm, Cuckoo Search,* dan *Flower Pollination Algorithm* untuk Optimasi Fungsi

Sebelum merumuskan *firefly algorithm*, *cuckoo search*, dan *flower pollination algorithm* untuk *multiple sequence alignment*, akan dirumuskan terlebih dahulu *firefly algorithm*, *cuckoo search*, dan *flower pollination algorithm* untuk optimasi fungsi. Yang akan dirumuskan antara lain adalah representasi solusi, cara membentuk solusi awal, modifikasi cara membentuk solusi baru dan modifikasi skema dari *firefly algorithm*, *cuckoo search*, dan *flower pollination algorithm*.

3.3 Pembuatan *Sequence*

Pembuatan *sequence* untuk *multiple sequence alignment* dilakukan agar *firefly algorithm*, *cuckoo search*, dan *flower pollination algorithm* dapat diimplementasikan untuk menyelesaikan *multiple sequence alignment* tersebut. Akan digunakan beberapa data *sequence* yang berbeda, baik dari segi banyaknya *sequence* yang digunakan maupun dari segi panjangnya *sequence*.

3.4 Perumusan *Modified Needleman-Wunsch Alignment*

Dalam penelitian ini akan dipaparkan metode baru untuk menggunakan metode progresif dengan memanfaatkan *Needleman-Wunsch alignment* untuk dua *sequence*. Nantinya metode baru itu akan disebut dengan *modified Needleman-Wunsch alignment*.

3.5 Perumusan *Firefly Algorithm*, *Cuckoo Search*, dan *Flower Pollination Algorithm* untuk *Multiple Sequence Alignment*

Firefly algorithm, *cuckoo search*, dan *flower pollination algorithm* yang telah dirumuskan untuk optimasi fungsi akan dimodifikasi agar bisa digunakan untuk menyelesaikan *multiple sequence alignment*. Beberapa yang akan dirumuskan antara lain adalah representasi solusi, cara membentuk solusi awal, dan cara membentuk solusi baru.

3.6 Pembuatan Simulasi

Masing-masing dari *firefly algorithm*, *cuckoo search*, dan *flower pollination algorithm* yang telah dirumuskan, baik untuk optimasi fungsi dan untuk *multiple sequence alignment*, akan disimulasikan dengan menggunakan bahasa pemrograman Java dalam NetBeans IDE 8.2. Nantinya *source code* dari simulasi yang telah dibuat akan ditampilkan di Lampiran.

3.7 Pembandingan Hasil

Pembandingan hasil dari *firefly algorithm*, *cuckoo search*, dan *flower pollination algorithm* dilakukan dengan memanfaatkan *sequence* dan simulasi yang telah dibuat. Perbandingan antara *firefly algorithm*, *cuckoo search*, dan *flower pollination algorithm* akan dilihat dari dua segi, yaitu skor *alignment* dan waktu komputasi.

3.8 Penarikan Kesimpulan

Penarikan kesimpulan dilakukan dengan memperhatikan hasil dan pembahasan yang telah diselesaikan pada tahap-tahap sebelumnya. Kesimpulan yang ditarik adalah mengenai apakah *firefly algorithm*, *cuckoo search*, dan *flower pollination algorithm* dapat digunakan untuk menyelesaikan *multiple sequence alignment* dengan baik dan bagaimana kemampuan dari algoritma-algoritma tersebut.

BAB IV PEMBAHASAN

4.1 *Firefly Algorithm, Cuckoo Search,* dan *Flower Pollination Algorithm* untuk Optimasi Fungsi

Tidak mudah untuk merumuskan *firefly algorithm, cuckoo search,* dan *flower pollination algorithm* yang dapat digunakan untuk menyelesaikan *multiple sequence alignment*. Hal ini dikarenakan *multiple sequence alignment* merupakan masalah yang kompleks. Terlebih lagi apabila belum ada dasar yang cukup dalam penggunaan *firefly algorithm, cuckoo search,* dan *flower pollination algorithm* untuk menyelesaikan permasalahan yang lebih sederhana.

Untuk menyediakan dasar pemahaman yang cukup untuk proses selanjutnya, pertama-tama akan dirumuskan *firefly algorithm, cuckoo search,* dan *flower pollination algorithm* untuk optimasi fungsi (mencari nilai maksimum atau minimum dari suatu fungsi), yang mana optimasi fungsi tersebut merupakan permasalahan yang paling sering digunakan untuk menguji kehandalan dari suatu *metaheuristic algorithm.* Nantinya, algoritma yang telah dirumuskan untuk optimasi fungsi akan dimodifikasi agar bisa digunakan untuk menyelesaikan *multiple sequence alignment*.

4.1.1 Representasi solusi untuk *Firefly Algorithm, Cuckoo Search,* dan *Flower Pollination Algorithm*

Setiap algoritma memiliki suatu representasi solusi untuk menyatakan suatu penyelesaian dari permasalahan. Apabila permasalahan yang akan diselesaikan berbeda, maka representasi solusi yang digunakan juga akan berbeda. Lebih lanjut lagi, satu permasalahan bisa menggunakan beberapa representasi yang berbeda pula.

Untuk sebuah permasalahan optimasi fungsi yang sederhana, biasanya sering digunakan representasi solusi biner, yaitu solusi yang berupa barisan sepanjang n dengan elemennya adalah ‘0’ dan ‘1’. Namun untuk permasalahan optimasi fungsi yang besar, representasi solusi tersebut menjadi sangat tidak efektif. Hal ini disebabkan karena representasi solusi akan berukuran sangat besar.

Representasi lain yang sering digunakan, dan yang paling sederhana, justru adalah solusi langsung yang berada dalam domain dari fungsi tersebut, yang biasanya disimpan dalam bentuk *array*. Misalkan domain dari suatu fungsi yang akan dicari penyelesaiannya adalah $\mathbb{R}^n$, maka representasi solusi yang digunakan adalah $[x_1 \quad x_2 \quad x_3 \quad \cdots \quad x_n]$ dimana $[x_1, x_2, x_3, \cdots, x_n]^T \in \mathbb{R}^n$.

Representasi solusi tersebut memiliki banyak keuntungan jika dibandingkan dengan representasi solusi biner, antara lain:

1. ukuran *array* yang digunakan jauh lebih kecil,
2. keakuratan solusi sangat jauh lebih baik,
3. jauh lebih mudah dalam mengoperasikan suatu solusi dengan solusi lainnya,
4. pembentukan solusi yang mudah,
5. dan tidak diperlukan suatu transformasi untuk merubah solusi.

4.1.1.1 Cara Membentuk Solusi

Solusi dibentuk dengan mengambil elemen dari domain fungsi secara acak dan menyimpannya dalam bentuk array. Biasanya domain fungsi yang digunakan adalah $a_i \leq x_i \leq b_i$ dengan $a_i, b_i \in \mathbb{R}$, untuk $1 \leq i \leq n$. Sehingga untuk mendapatkan solusi acak, dapat digunakan

$$x_i = r_i(b_i - a_i) + a_i \tag{4.1}$$

dimana r_i adalah bilangan acak yang diambil dari distribusi Uniform $U(0, 1)$.

Pseudo code yang digunakan untuk membentuk solusi ditampilkan dalam Gambar 4.1.

```
create_solution
    input  : n (dimensi solusi)
             a (array berukuran n)
             b (array berukuran n)

    output : x (solusi berukuran n)

    for i = 1 to n
        x(i) = random(0,1)*(b(i)-a(i)) + a(i)
    return x
```

Gambar 4.1 *Pseudo Code* untuk Membentuk Solusi

Dengan persamaan (4.1), solusi yang terbentuk akan selalu berada dalam domain fungsi yang ditentukan. Hal inilah yang disebut dengan pembentukan solusi yang mudah dan tidak diperlukan suatu transformasi untuk merubah solusi. Ukuran *array* yang digunakan akan memiliki ukuran yang sama dengan dimensi dari permasalahan, yaitu n.

4.1.2 Modifikasi *Firefly Algorithm*, *Cuckoo Search*, dan *Flower Pollination Algorithm*

Perlu dilakukan modifikasi pada *firefly algorithm*, *cuckoo search*, dan *flower pollination algorithm* agar dapat digunakan untuk menyelesaikan masalah optimasi fungsi dengan hasil yang semakin baik. Persamaan-persamaan yang telah dituliskan dalam (2.8), (2.9), (2.10), (2.11), dan (2.12) juga perlu dimodifikasi. Setiap suku yang dianggap terlalu sulit akan dihilangkan dan diganti dengan suku yang lebih mudah. Hal ini dilakukan karena tujuan utama dalam penelitian ini bukanlah untuk menyelesaikan optimasi fungsi, melainkan untuk menyelesaikan *multiple sequence alignment*.

4.1.3 *Firefly Algorithm* untuk Optimasi Fungsi

Secara umum, *firefly algorithm* adalah algoritma yang jauh lebih sederhana daripada algoritma lainnya. Algoritma yang lain biasanya menggunakan lebih dari satu macam mekanisme untuk mendapatkan solusi baru, namun *firefly algorithm* hanya menggunakan satu cara. Namun hal tersebut tidak menjadikannya kalah dari algoritma yang lain. Digabungkannya pergerakan acak dan pergerakan lokal dalam satu mekanisme membuat *firefly algorithm* dapat menemukan solusi yang baik. Dan apabila diamati lebih lanjut mengenai pergerakan *firefly* dalam (2.8), maka pergerakan lokal akan berubah menjadi pergerakan global saat $x_j^t = x_*^t$, dengan $j \in \{1, 2, \dots, n\}$, adalah solusi terbaik dalam populasi *firefly*.

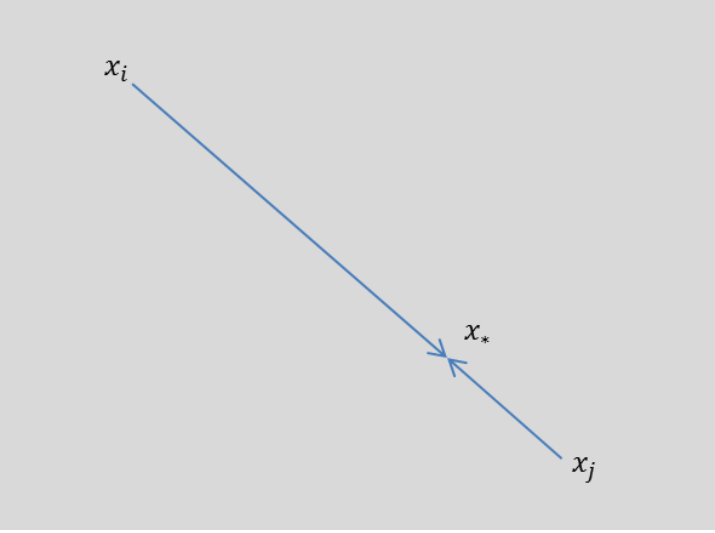
Perhatikan bahwa karena $\gamma > 0$ dan $r_{ij} > 0$ maka $\gamma r_{ij}^2 > 0$ dan $e^{-\gamma r_{ij}^2} \in (0, 1)$. Terlebih lagi, pada awalnya r_{ij} akan bernilai cukup besar karena *firefly*-*firefly* masih terletak berjauhan. Apabila r_{ij} cukup besar maka $e^{-\gamma r_{ij}^2}$ akan bernilai mendekati 0. Hal ini menyebabkan $I_0 e^{-\gamma r_{ij}^2}$ juga akan mendekati 0 dan

$I_0 e^{-\gamma r_{ij}^2}\left(x_j^t - x_i^t\right)$ menjadi tidak berpengaruh dalam perhitungan x_i^{t+1}. Oleh karena itu suku tersebut diganti dengan $a_i^t \in (0,1)$, bilangan acak antara 0 sampai 1. a_i^t tersebut akan berpengaruh dalam perhitungan x_i^{t+1} pada semua kasus.

Parameter α dalam (2.8), yang merupakan parameter untuk pergerakan acak, dimodifikasi menjadi $b_i^t(x_*^t - x_i^t)$, dengan $b_i^t \in (-1,1)$. Manfaat dari modifikasi ini adalah tidak perlu dipikirkannya pemilihan nilai untuk α. Hal ini karena pemilihan α harus mempertimbangkan permasalahan yang dikerjakan. Setiap permasalahan menggunakan α yang berbeda-beda dan perlu ada pengalaman untuk bisa menentukan nilai α yang baik dalam permasalahan tersebut. $b_i^t(x_*^t - x_i^t)$ akan dengan sendirinya menyesuaikan besarnya pergerakan acak dari *firefly*. Pemilihan $b_i^t \in (-1,1)$ dimaksudkan agar pergerakan acak tidak selalu memperbesar nilai dari x_i^{t+1}, melainkan juga bisa memperkecilnya. Sehingga pergerakan dari *firefly* dalam (2.8) berubah menjadi

$$x_i^{t+1} = x_i^t + a_i^t\left(x_j^t - x_i^t\right) + b_i^t \epsilon_i^t (x_*^t - x_i^t). \tag{4.2}$$

Selanjutnya perhatikan Gambar 4.2 berikut.



Gambar 4.2 Ilustrasi dari Dua Solusi Berbeda yang Dapat Bergerak ke Arah yang Lebih Baik

Dalam Gambar 4.2, diilustrasikan bahwa solusi optimum dari permasalahan adalah x_*. Diilustrasikan pula terdapat dua solusi berbeda yaitu x_i dan x_j. Dapat

diasumsikan bahwa x_j adalah solusi yang lebih baik daripada x_i karena x_j lebih dekat dengan x_*.

Jika kita menggunakan *firefly algorithm* yang telah dijelaskan sebelumnya, maka x_i akan bergerak menuju x_j yang artinya x_i akan semakin dekat dengan x_*. Namun x_j tidak dapat bergerak menuju x_i mengingat bahwa x_j adalah solusi yang lebih baik dari x_i. Padahal sangat mungkin didapatkan solusi yang lebih baik saat x_j bergerak menuju x_i. Dan pada dasarnya, dari manapun solusi bergerak, solusi tersebut selalu dapat menuju ke arah yang lebih baik. Namun tidak menutup kemungkinan bahwa solusi tersebut akan bergerak ke arah yang lebih buruk. Oleh karena itu akan lebih baik apabila solusi baru hanya akan diterima saat ia lebih baik daripada solusi sebelumnya. *Pseudo code* dari *firefly algorithm* yang telah dirumuskan ditampilkan dalam Gambar 4.3.

```
firefly_algorithm
    input  : n (banyak solusi)
             x (solusi berukuran n)
             toleransi
             max_iterasi
    output : x* (solusi terbaik)

    iterasi = 0
    x* = best(x)
    while iterasi < max_iterasi and f(x*) > toleransi
        for i = 1 to n
            for j = 1 to n
                hitung x_baru(i) dengan (4.2)
                if f(x_baru(i)) > f(x(i))
                    x(i) = x_baru(i)
        x* = best(x)
        iterasi = iterasi + 1
    return x*
```

Gambar 4.3 *Pseudo Code* untuk *Firefly Algorithm*

Firefly algorithm dapat dijalankan sampai sebanyak m iterasi seperti yang ditunjukkan dalam Gambar 4.2. Hal tersebut dapat diganti dengan sampai didapatkannya $f(x_*) < T$, dimana T adalah batas toleransi yang diinginkan. Dapat pula apabila digunakan keduanya, yaitu *firefly algorithm* terus dijalankan sampai sebanyak m iterasi atau sampai didapatkan $f(x_*) < T$.

4.1.4 *Cuckoo Search* untuk Optimasi Fungsi

Pergerakan lokal tetap dilakukan dengan (2.9), hanya saja *step size* s dirubah menjadi r_i^t, yaitu bilangan acak yang diambil dari distribusi Uniform $U(0,1)$.

$$x_i^{t+1} = x_i^t + r_i^t\left(x_j^t - x_k^t\right) \tag{4.3}$$

Hal ini karena s yang bernilai konstan terkadang tidak bermanfaat pada kasus-kasus tertentu.

Di sisi lain, pergerakan Levy dalam (2.10) cukup susah untuk digunakan. Karena perlu langkah-langkah yang sangat banyak dalam menggunakan pergerakan Levy tersebut. Oleh karena itu (2.10) dirubah menjadi

$$x_i^{t+1} = x_i^t + b_i^t c_i^t (x_*^t - x_i^t) \tag{4.4}$$

dimana s adalah *step size*, b_i^t adalah bilangan acak yang diambil dari distribusi Uniform $U(0,1)$, dan c_i^t adalah bilangan acak yang diambil dari distribusi Uniform $U(-1,1)$.

```
cuckoo_search
    input  : n (banyak solusi)
             x (solusi berukuran n)
             toleransi
             max_iterasi
             p_a
    output : x* (solusi terbaik)

    iterasi = 0
    x* = best(x)
    while iterasi < max_iterasi and f(x*) > toleransi
        for i = 1 to n
            hitung x_baru(i) dengan (4.4)
        x = elitism_replacement_with_filtration(x,x_baru)
        for i = 1 to n
            r = random(0,1)
            if r < p_a
                hitung x_baru(i) dengan (4.3)
            if f(x_baru(i)) > f(x(i))
                x(i) = x_baru(i)
        x* = best(x)
        iterasi = iterasi + 1
    return x*
```

Gambar 4.4 *Pseudo Code* untuk *Cuckoo Search*

Dalam *cuckoo search* yang dijelaskan sebelumnya, pergerakan global akan diterima apabila solusi baru yang didapatkan lebih baik daripada solusi sebelumnya. Mekanisme tersebut akan diganti dengan *elitism replacement with filtration*. Sehingga solusi tersebut akan mengambil semua solusi terbaik, entah itu dari solusi yang baru atau yang lama.

Kemudian, solusi baru yang diciptakan dengan dengan pergerakan lokal, tidak akan benar-benar langsung mengganti solusi lama. Solusi baru tersebut akan diterima hanya apabila ia lebih baik daripada solusi sebelumnya. Parameter p_a yang digunakan dalam algoritma ini adalah $1/2$.

Berdasarkan apa yang telah dipaparkan, *pseudo code* dari *cuckoo search* yang diusulkan adalah seperti yang ditampilkan dalam Gambar 4.5.

```
flower_pollination_algorithm
    input  : n (banyak solusi)
             x (solusi berukuran n)
             toleransi
             max_iterasi
             p
    output : x* (solusi terbaik)

    iterasi = 0
    x* = best(x)
    while iterasi < max_iterasi and f(x*) > toleransi
        for i = 1 to n
            r = random(0,1)
            if r < p
                hitung x_baru(i) dengan (4.6)
            else
                hitung x_baru(i) dengan (4.5)
        x = elitism_replacement_with_filtration(x,x_baru)
        x* = best(x)
        iterasi = iterasi + 1
    return x*
```

Gambar 4.5 *Pseudo Code* untuk *Flower Pollination Algorithm*

4.1.5 *Flower Pollination Algorithm* untuk Optimasi Fungsi

Pergerakan lokal untuk *flower pollination algorithm* dilakukan dengan cara yang hampir sama dengan pergerakan dari *firefly algorithm* dalam (4.2). Hanya

saja suku $x_j^t - x_i^t$ dirubah menjadi $x_j^t - x_k^t$ sehingga pergerakan lokalnya dinyatakan dengan

$$x_i^{t+1} = x_i^t + a_i^t\left(x_j^t - x_k^t\right) + b_i^t \epsilon_i^t (x_*^t - x_i^t). \tag{4.5}$$

Nilai ε yang konstan diganti dengan a_i^t dan ditambahkan sedikit pergerakan acak $b_i^t \epsilon_i^t (x_*^t - x_i^t)$.

Dan pergerakan global untuk *flower pollination algorithm* dilakukan dengan

$$x_i^{t+1} = x_i^t + a_i^t (x_*^t - x_i^t) \tag{4.6}$$

dimana a_i^t adalah bilangan acak yang diambil dari distribusi Uniform $U(0,1)$. Hal ini dilakukan karena pergerakan Levy dalam (2.11) cukup susah untuk digunakan. Karena perlu langkah-langkah yang sangat banyak dalam menggunakan pergerakan Levy tersebut. Dan nilai γ yang konstan diganti dengan a_i^t.

Pergerakan global dan pergerakan lokal terjadi dengan peluang p dan $1 - p$. Karena terlalu sering melakukan pergerakan global akan menyebabkan cepat konvergennya suatu populasi, maka dipilih $p = 0.1$. Sehingga pergerakan global memiliki peluang terjadi 0.1 dan pergerakan lokal memiliki peluang terjadi 0.9. Berdasarkan apa yang telah dipaparkan, *pseudo code* dari *flower pollination algorithm* yang diusulkan adalah seperti yang ditampilkan dalam Gambar 4.5.

4.1.6 Fungsi yang Dipakai untuk Pengujian

Untuk menguji *firefly algorithm*, *cuckoo search*, dan *flower pollination algorithm* yang telah dipaparkan, digunakan fungsi

$$f(x) = \sum_{i=1}^{n} (x_i - 50)^2 \tag{4.7}$$

dengan domain $100 \leq x_i \leq 100$ untuk $1 \leq i \leq n$. Nantinya akan digunakan beberapa nilai n dalam pengujian. Semakin besar n yang dipakai, maka akan semakin susah proses pengerjaannya. Perlu diingat pula bahwa solusi minimum dari permasalahan tersebut adalah $f(x) = 0$ yaitu saat $x_1 = x_2 = \cdots = x_n = 50$.

```
solution_fitness
    input  : n (dimensi solusi)
             x (solusi berukuran n)
    output : f (fitness dari solusi)

    f = 0;
    for i = 1 to n
        f = f + (x(i)-1)^2
    return x
```

Gambar 4.6 *Pseudo Code* untuk Menghitung *Fitness* dari Solusi

Dengan dipakainya fungsi tersebut, maka cara yang digunakan untuk menghitung *fitness* dari solusi adalah seperti yang ditampilkan dalam Gambar 4.6.

4.1.7 Perbandingan Hasil Algoritma

Untuk mengetahui perbandingan dari *firefly algorithm*, *cuckoo search,* dan *flower pollination*, digunakan (4.7) sebagai fungsi *fitness* dengan beberapa nilai n yaitu 10, 25, 50, 100, 250, dan 500. Kondisi berhentinya algoritma yang digunakan adalah saat fungsi *fitness* lebih kecil dari toleransi T, yaitu 10^{-10} dan 10^{-100}, sehingga perbandingan hanya mempertimbangkan waktu komputasi dari masing-masing algoritma. Dalam Tabel 4.1, beberapa nilai dari *firefly algorithm* yang asli tidak ditampilkan karena dalam beberapa kasus tidak dapat menemukan solusi yang *fitness*-nya lebih kecil dari T.

Tabel 4.1 Perbandingan Algoritma untuk Optimasi Fungsi

n	T	Waktu Komputasi					
		Firefly Algorithm	*Modified Firefly Algorithm*	*Cuckoo Search*	*Modified Cuckoo Search*	*Flower Pollination Algorithm*	*Modified Flower Pollination Algorithm*
10	10^{-10}	0.559	**0.009**	0.111	0.013	0.013	0.011
10	10^{-100}	1.824	0.009	0.092	**0.007**	0.014	0.009
25	10^{-10}	1.213	0.015	0.197	**0.010**	0.026	0.011
25	10^{-100}	-	0.027	0.219	**0.019**	0.055	0.027
50	10^{-10}	2.419	0.035	0.389	**0.020**	0.077	0.030
50	10^{-100}	-	0.087	0.369	**0.047**	0.165	0.072
100	10^{-10}	4.951	0.103	0.719	**0.055**	0.212	0.085
100	10^{-100}	-	0.186	0.712	**0.108**	0.447	0.193
250	10^{-10}	12.638	0.336	1.651	**0.193**	0.801	0.325
250	10^{-100}	-	0.717	1.648	**0.363**	1.759	0.707
500	10^{-10}	26.718	0.884	3.249	**0.557**	2.184	1.150
500	10^{-100}	-	1.877	3.274	**1.041**	4.601	1.959

Tampak bahwa kemampuan dari *modified cuckoo search* lebih baik dari *modified firefly algorithm* dan *modified flower pollination algorithm*. Namun *modified firefly algorithm* dan *modified flower pollination algorithm* tidak kalah jauh dari *modified cuckoo search*. Kemampuan dari *modified firefly algorithm* hampir sebanding dengan *modified flower pollination algorithm*. Secara umum, *modified firefly algorithm*, *modified cuckoo search*, dan *modified flower pollination algorithm* dapat digunakan untuk menyelesaikan optimasi fungsi dengan waktu komputasi yang cukup singkat, walaupun n diperbesar dan T sangat kecil. Modifikasi yang dilakukan juga membuat algoritma-algoritma tersebut lebih baik daripada algoritma-algoritma aslinya.

4.2 Pembangkitan *Sequence*

Untuk membangkitkan *sequence-sequence* yang akan di-*alignment* dan digunakan sebagai *input* untuk *firefly algorithm*, *cuckoo search,* dan *flower pollination algorithm*, perlu dibentuk sebuah *parent* terlebih dahulu. Kemudian *sequence-sequence* baru dapat dibentuk dengan mengenakan beberapa mutasi pada *parent* tersebut.

```
generate_parent
    input  : l (panjang parent)
    output : parent

    parent = ""
    for i = 1 to l
        r ← random(0,1)
        if r < 0.25
            parent = concatenate(parent, "A")
        if else r < 0.5
            parent = concatenate(parent, "C")
        if else r < 0.75
            parent = concatenate(parent, "G")
        else
            parent = concatenate(parent, "T")
    return parent
```

Gambar 4.7 *Pseudo Code* untuk Membangkitkan *Parent*

4.2.1 Membentuk *Parent*

Misalkan akan dibentuk sebuah *parent*, yaitu *sequence* DNA yang tersusun dari l *nucleotide*. Pembentukan *parent* tersebut dilakukan dengan membangkitkan l *nucleotide* dan kemudian menyusunnya. Setiap *nucleotide* dapat dibangkitkan dengan memilih secara acak suatu elemen dari himpunan $\{A, C, G, T\}$. Masing-masing elemen dari himpunan tersebut memiliki peluang $1/4$ untuk terpilih. Secara umum cara pembentukan *parent* tersebut ditampilkan dalam *pseudo code* berikut.

4.2.2 Membentuk *Child*

Apabila sudah terbentuk sebuah *parent*, maka dapat dibentuk *sequence* baru dengan menggunakan beberapa mutasi. Seperti yang telah dijelaskan, terdapat empat tipe mutasi yang dapat terjadi. Setiap tipe mutasi tersebut dapat terjadi pada setiap *nucleotide* dari *parent*.

Misalkan bahwa peluang terjadinya setiap tipe mutasi adalah 0.1. Maka total peluang terjadinya mutasi pada setiap *nucleotide* dari *parent* adalah 0.4. Sisanya, yaitu sebesar 0.6, adalah peluang tidak terjadinya mutasi. Misalkan *child* menyatakan *sequence* baru yang dibentuk dengan menjalankan beberapa mutasi pada *parent*. Secara umum *child* dapat dibentuk dengan *pseudo code* yang ditampilkan pada Gambar 4.8.

Apabila ingin dibentuk beberapa *child*, misalkan sebanyak n, maka algoritma tersebut hanya tinggal dijalankan sebanyak n kali. Selanjutnya, n *child* tersebut akan di-*alignment* dan digunakan sebagai *input* untuk *firefly algorithm*, *cuckoo search*, dan *flower pollination algorithm*.

```
generate_child
    input  : l (panjang parent)
             parent
    output : child

    child = ""
    for i = 1 to l
        r1 = random(0,1)
        if r < 0.1
            r2 = random(0,1)
            if r2 < 0.25
                child = concatenate(child, "A")
            if else r2 < 0.5
                child = concatenate(child, "C")
            if else r2 < 0.75
                child = concatenate(child, "G")
            else
                child = concatenate(child, "T")
        if else r1 < 0.2
            child = concatenate(child, parent(i+1), parent(i))
            i = i + 1
        if else r1 < 0.3
            r2 = random(0,1)
            if r2 < 0.125
                child = concatenate(child, "A", parent(i))
            if else r2 < 0.25
                child = concatenate(child, "C", parent(i))
            if else r2 < 0.375
                child = concatenate(child, "G", parent(i))
            if else r2 < 0.5
                child = concatenate(child, "T", parent(i))
            if else r2 < 0.625
                child = concatenate(child, parent(i), "A")
            if else r2 < 0.75
                child = concatenate(child, parent(i), "C")
            if else r2 < 0.875
                child = concatenate(child, parent(i), "G")
            else
                child = concatenate(child, parent(i), "T")
        if else r1 < 0.4
            child = child
        else
            child = concatenate(child, parent(i))
    return child
```

Gambar 4.8 *Pseudo Code* untuk Membangkitkan *Child*

4.2.3 *Sequence* yang Dipakai untuk Pengujian

Untuk melakukan *alignment*, diperlukan beberapa *sequence* sebagai *input*. Oleh karena itu dibangkitkan *sequence-sequence* dengan menggunakan *pseudo code* yang telah dijelaskan. Pertama-tama dibentuk beberapa *parent*, panjangnya bervariasi mulai dari 10 sampai 500. Dari setiap *parent* kemudian dibentuk 5 *sequence* baru.

4.3 *Modified* Needleman-Wunsch *Alignment*

Dalam bagian ini, penulis memaparkan metode baru dengan memanfaatkan Needleman-Wuncsh *alignment* untuk dua *sequence*.

Misalkan kita mempunyai *alignment*

G G C A – A

G – C A C A

G G – A C A

dan misalkan kita mempunyai *sequence* baru yaitu

T T T C A C A.

Dari pada kita memulai dari awal proses *alignment* lebih baik kita mencari cara baru untuk melakukan *alignment* antara *alignment* pertama dan *sequence* yang baru. Untuk melakukan hal tersebut, penulis melakukan modifikasi pada matriks skor dari Needleman-Wuncsh *alignment* untuk dua *sequence*. Contoh modifikasi untuk contoh kasus di atas ditampilkan dalam Tabel 4.2.

Apabila dilakukan *back track*, akan didapatkan *alignment* baru yaitu

– G G C A – A

– G – C A C A

– G G – A C A

T T T C A C A

G – G C A – A

G – – C A C A

G – G – A C A

T T T C A C A

G G – C A – A

G – – C A C A

G G – – A C A

T T T C A C A

Tabel 4.2 Modifikasi Matriks Skor

M	–	T	G	T	C	A	C	A
– – –	0	-6	-12	-18	-24	-30	-36	-42
G G G	-3	3	-3	-9	-15	-21	-27	-33
G – G	-10	-4	-2	-8	-14	-20	-26	-32
C C –	-17	-11	-9	-7	-11	-17	-23	-29
A A A	-20	-14	-8	-6	-4	-5	-11	-17
– C C	-27	-21	-15	-13	-9	-9	-8	-14
A A A	-30	-24	-18	-12	-10	-3	-6	-2

Dengan cara yang serupa, apabila dilakukan *alignment* antara

G G C A – A

G – C A C A

G G – A C A

dan

T T T C A C A

– T G C – C A

maka matriks skor yang digunakan adalah seperti yang ditampilkan dalam Tabel 4.3. Proses *back track* dari hasil tersebut serupa dengan *backtrack* yang biasanya.

Metode baru Needleman-Wuncsh *alignment* dikerjakan dengan pertama-tama mencari sebuah *sequence star*, seperti yang dilakukan dalam *star alignment*. Kemudian menggunakan Needleman-Wuncsh *alignment* untuk dua *sequence* untuk mendapatkan *alignment* yang pertama. Selanjutnya *alignment* yang didapat, digabung dengan *sequence* ketiga dengan metode baru Needleman-Wunsch *alignment*. Selanjutnya *alignment* yang didapat, digabung dengan *sequence* keempat dengan metode baru Needleman-Wunsch *alignment*. Begitu seterusnya hingga semua *sequence* selesai diproses.

Tabel 4.3 Modifikasi Matriks Skor

M	- -	T -	T T	T G	C C	A -	C C	A A
- - -								
G G G								
G - G								
C C -								
A A A								
- C C								
A A A								

Apabila akan dilakukan *alignment* untuk k *sequence* yang rata-rata panjangnya *sequence* adalah n, maka Needleman-Wunsch *alignment* membutuhkan n^k perhitungan. Sedangkan *modified* Needleman-Wunsch *alignment* hanya membutuhkan $n^2(k-1)$ perhitungan. Untuk k yang semakin besar, tentu *modified* Needleman-Wunsch *alignment* memiliki waktu komputasi yang lebih kecil daripada Needleman-Wunsch *alignment* yang asli.

```
modified_needleman_wunsch_alignment
    input  : a (array string[])
             n (ukuran dari a)

    output : s

    score = needleman_wunsch_pairs_score(a)
    for i = 1 to n
        row_score(i) = 0
        for j = 1 to n
            row_score(i) = row_score(i) + score(i)(j)

    star = 0
    max = row_score(1)
    for i = 2 to n
        if row_score(i) > max
            max = row_score(i)
            star = i

    if star ~= 1
        b = a(star)
        a(star) = a(1)
        a(1) = b

    s = needleman_wunsch_alignment(a(1), a(2))
    for i = 3 to n
        s = needleman_wunsch_alignment(s, a(i))
    }

    return s
```

Gambar 4.9 *Pseudo Code* untuk *Modified Needleman-Wunsch Alignment*

```
needleman_wunsch_alignment
    input  : a (array string[])
             n (ukuran dari a)

    output : s

    score = needleman_wunsch_pairs_score(a)
    for i = 1 to n
        row_score(i) = 0
        for j = 1 to n
            row_score(i) = row_score(i) + score(i)(j)

    star = 0
    max = row_score(1)
    for i = 2 to n
        if row_score(i) > max
            max = row_score(i)
            star = i

    if star ~= 1
        b = a(star)
        a(star) = a(1)
        a(1) = b

    s = needleman_wunsch_alignment(a(1), a(2))
    for i = 3 to n
        s = needleman_wunsch_alignment(s, a(i))
    }

    return s
```

Gambar 4.10 *Pseudo Code* untuk *Needleman-Wunsch Alignment*

4.3.1 Perbandingan Needleman-Wunsch *Alignment, Star Alignment,* dan *Modified* Needleman-Wunsch *Alignment*

Untuk menguji seberapa baik *modified Needleman-Wunsch alignment* yang telah dipaparkan, dilakukan pengujian dengan memanfaatkan *sequence-sequence* yang sebelumnya telah dibangkitkan.

Perbandingan dari *Needleman-Wunsch alignment*, *star alignment*, dan *modified Needleman-Wunsch alignment* ditampilkan dalam Tabel 4.4, Tabel 4.5, dan Tabel 4.6. Skor terbaik dari *star alignment*, dan *modified Needleman-Wunsch alignment* ditampilkan dengan huruf tebal.

Tabel 4.4 Perbandingan Hasil dari *Needleman-Wunsch Alignment*, *Star Alignment*, dan *Modified Needleman-Wunsch Alignment* untuk 3 *Sequence*

l	*Needleman-Wuncsh Alignment*		*Star Alignment*		*Modified Needleman-Wunsch Alignment*	
	Skor	Waktu	Skor	Waktu	Skor	Waktu
10	15	0.003	**15**	0.000	**15**	0.001
20	13	0.012	**12**	0.001	**12**	0.002
30	41	0.042	38	0.002	**40**	0.004
40	50	0.040	48	0.003	**50**	0.006
50	64	0.028	**63**	0.004	**63**	0.006
60	77	0.019	76	0.005	**77**	0.011
70	83	0.155	76	0.006	**80**	0.012
80	116	0.034	110	0.005	**115**	0.008
90	94	0.028	89	0.002	**90**	0.009
100	124	0.054	114	0.001	**118**	0.007
110	135	0.059	124	0.002	**132**	0.008
120	149	0.093	131	0.001	**143**	0.009
130	170	0.112	151	0.001	**161**	0.011
140	187	0.123	178	0.002	**183**	0.016
150	199	0.170	187	0.001	**198**	0.014
160	213	0.191	182	0.001	**201**	0.010
170	174	0.243	146	0.002	**168**	0.019
180	190	0.236	146	0.002	**176**	0.009
190	227	0.291	198	0.002	**216**	0.010
200	257	0.363	209	0.002	**243**	0.011
210	225	0.428	204	0.002	**217**	0.011
220	267	0.472	248	0.002	**260**	0.013
230	261	0.568	233	0.002	**252**	0.014
240	305	0.643	284	0.002	**299**	0.015
250	288	0.719	248	0.002	**282**	0.017
260	306	0.798	266	0.002	**297**	0.018
270	348	0.876	332	0.003	**341**	0.019
280	329	0.962	287	0.004	**316**	0.021
290	343	1.090	315	0.004	**333**	0.022
300	364	1.199	317	0.004	**357**	0.024
310	380	1.313	337	0.004	**365**	0.025
320	385	1.347	345	0.004	**366**	0.027
330	407	1.656	362	0.005	**394**	0.029
340	382	1.627	347	0.005	**361**	0.031
350	410	1.828	361	0.005	**399**	0.029
360	429	2.202	384	0.005	**416**	0.034
370	462	2.325	406	0.004	**447**	0.037
380	470	2.427	417	0.006	**459**	0.034
390	465	2.751	416	0.006	**446**	0.038
400	483	2.835	433	0.006	**468**	0.041
410	478	3.153	410	0.006	**457**	0.049
420	495	3.377	440	0.006	**474**	0.050
430	485	3.534	427	0.006	**467**	0.048
440	503	3.834	463	0.007	**486**	0.048
450	556	4.064	495	0.007	**533**	0.050
460	587	4.452	520	0.008	**575**	0.052
470	570	4.726	505	0.007	**555**	0.057
480	576	5.037	492	0.008	**559**	0.062
490	548	5.107	457	0.008	**517**	0.063
500	634	5.808	573	0.010	**618**	0.061

Tabel 4.5 Perbandingan Hasil dari *Star Alignment* dan *Modified Needleman-Wunsch Alignment* untuk 4 *Sequence*

l	*Star Alignment*		*Modified Needleman-Wunsch Alignment*	
	Skor	Waktu	Skor	Waktu
10	12	0.003	**20**	0.002
20	23	0.001	**26**	0.005
30	59	0.002	**63**	0.007
40	72	0.004	**77**	0.011
50	88	0.006	**104**	0.015
60	125	0.004	**127**	0.012
70	126	0.006	**144**	0.011
80	208	0.004	**219**	0.018
90	132	0.003	**154**	0.039
100	204	0.003	**229**	0.042
110	243	0.001	**263**	0.047
120	234	0.001	**276**	0.038
130	244	0.003	**283**	0.021
140	311	0.002	**323**	0.018
150	325	0.002	**348**	0.017
160	326	0.002	**382**	0.022
170	304	0.003	**335**	0.025
180	299	0.003	**344**	0.034
190	396	0.004	**420**	0.031
200	373	0.003	**447**	0.027
210	365	0.003	**430**	0.033
220	453	0.004	**492**	0.034
230	356	0.004	**437**	0.044
240	480	0.006	**516**	0.061
250	433	0.007	**530**	0.052
260	501	0.006	**575**	0.048
270	589	0.006	**615**	0.052
280	538	0.008	**607**	0.057
290	525	0.006	**601**	0.065
300	629	0.007	**661**	0.063
310	609	0.007	**674**	0.066
320	592	0.024	**679**	0.074
330	660	0.009	**765**	0.095
340	528	0.008	**693**	0.073
350	659	0.008	**727**	0.082
360	618	0.008	**726**	0.084
370	726	0.009	**850**	0.109
380	727	0.010	**803**	0.095
390	699	0.011	**821**	0.117
400	788	0.012	**857**	0.105
410	792	0.056	**895**	0.113
420	858	0.011	**945**	0.109
430	791	0.011	**912**	0.112
440	765	0.012	**934**	0.124
450	870	0.012	**999**	0.129
460	965	0.013	**1090**	0.136
470	907	0.014	**1020**	0.139
480	924	0.013	**1053**	0.137
490	957	0.013	**1045**	0.143
500	990	0.015	**1114**	0.139

Tabel 4.6 Perbandingan Hasil dari *Star Alignment* dan *Modified Needleman-Wunsch Alignment* untuk 5 *Sequence*

l	*Star Alignment*		*Modified Needleman-Wunsch Alignment*	
	Skor	Waktu	Skor	Waktu
10	17	0.002	**31**	0.003
20	16	0.002	**23**	0.006
30	111	0.004	**122**	0.010
40	83	0.007	**105**	0.018
50	122	0.004	**160**	0.014
60	188	0.006	**196**	0.014
70	170	0.006	**226**	0.025
80	303	0.003	**332**	0.054
90	227	0.003	**291**	0.078
100	286	0.002	**337**	0.078
110	343	0.003	**393**	0.021
120	363	0.003	**437**	0.019
130	372	0.002	**431**	0.035
140	409	0.003	**482**	0.027
150	482	0.004	**526**	0.043
160	518	0.005	**591**	0.053
170	470	0.004	**538**	0.040
180	474	0.004	**590**	0.046
190	629	0.005	**694**	0.048
200	593	0.007	**718**	0.071
210	587	0.010	**680**	0.059
220	639	0.005	**796**	0.054
230	655	0.006	**745**	0.073
240	754	0.007	**845**	0.065
250	717	0.006	**847**	0.093
260	751	0.011	**882**	0.114
270	898	0.008	**968**	0.081
280	809	0.008	**984**	0.088
290	796	0.009	**926**	0.116
300	928	0.010	**1014**	0.119
310	925	0.011	**1066**	0.123
320	848	0.010	**1066**	0.113
330	975	0.011	**1153**	0.149
340	787	0.012	**1071**	0.131
350	988	0.012	**1160**	0.152
360	956	0.013	**1179**	0.142
370	1195	0.014	**1375**	0.147
380	1095	0.015	**1263**	0.155
390	1016	0.015	**1243**	0.200
400	1142	0.016	**1344**	0.185
410	1178	0.018	**1362**	0.204
420	1325	0.035	**1487**	0.200
430	1165	0.017	**1414**	0.185
440	1142	0.020	**1468**	0.229
450	1310	0.019	**1564**	0.233
460	1444	0.020	**1655**	0.220
470	1319	0.021	**1554**	0.251
480	1419	0.022	**1659**	0.244
490	1479	0.022	**1671**	0.253
500	1597	0.024	**1812**	0.258

Secara kesuluruhan terdapat 150 kasus yang digunakan untuk melakukan uji coba, dan dari keseluruhan kasus tersebut *modified Needleman-Wunsch alignment* memiliki skor yang sama dengan *star alignment* pada 3 kasus. Dan pada kasus yang lain, selalu lebih baik dari *star alignment*.

Modified Needleman-Wunsch alignment memiliki skor yang tidak jauh berbeda dari *Needleman-Wunsch alignment* yang asli pada pengujian dengan tiga *sequence*.

Total waktu komputasi dari *modified Needleman-Wunsch alignment* adalah 9.693 detik, sedang untuk *star alignment* adalah 1.094 detik. Sehingga rata-rata *modified Needleman-Wunsch alignment* membutuhkan waktu komputasi sebesar 8.860 kali dari waktu komputasi *star alignment*.

4.4 *Firefly Algorithm, Cuckoo Search,* dan *Flower Pollination Algorithm* untuk *Multiple Sequence Alignment*

Secara umum, *firefly algorithm*, *cuckoo search*, dan *flower pollination algorithm* untuk *multiple sequence alignment* menggunakan langkah-langkah yang hampir sama dengan yang digunakan untuk optimasi fungsi, yaitu seperti yang telah dijelaskan dalam Gambar 4.3, Gambar 4.4, dan Gambar 4.5. Yang berbeda adalah representasi solusi dan cara untuk membentuk solusi baru.

4.4.1 Representasi Solusi

Representasi solusi yang digunakan untuk *firefly algorithm, cuckoo search,* dan *flower pollination algorithm* adalah solusi dari *multiple sequence alignment* itu sendiri. Artinya, representasi solusinya adalah berupa *sequence-sequence* awal yang disisipi gap "−" sehingga panjang setiap *sequence* menjadi sama. Representasi solusi ini lebih mudah diolah dan tidak diperlukan suatu transformasi lain yang akan memakan banyak waktu.

4.4.1.1 Cara Membentuk Solusi

Solusi bisa dibentuk dengan berbagai cara. Secara umum solusi dari *multiple sequence alignment* dibuat dengan menempatkan gap "−" secara acak sampai didapati panjang setiap *sequence* menjadi sama. Cara tersebut adalah cara

konvensional yang memiliki banyak kekurangan. Hal ini dikarenakan *multiple sequence alignment* merupakan masalah yang kompleks.

Oleh karena itu, solusi awal untuk algoritma bisa dibuat dengan metode-metode sederhana untuk *multiple sequence alignment*, misalnya *star alignment*. Dalam penelitian ini, solusi awal dibentuk secara acak dan juga dengan menggunakan metode baru Needleman Wunsch *alignment* yang telah dipaparkan sebelumnya. Kemudian, *firefly algorithm*, *cuckoo search*, dan *flower pollination algorithm* akan digunakan untuk meningkatkan solusi-solusi awal tersebut dengan beberapa cara tertentu.

4.4.2 Cara Membentuk Solusi Baru

Dalam bagian ini akan diperkenalkan dua cara untuk membentuk solusi baru. Cara pertama didasari dari mungkinnya dilakukan penggabungan antara dua kolom dari suatu *alignment* apabila setiap *nucleotide* yang terdapat dalam suatu baris bersebelahan dengan suatu gap pada baris yang sama. Misalkan dipunyai suatu solusi

```
- G G - A - A
T G - C A C A
- G G - A C A
T T - C A C A
```

yang memiliki skor -9. Apabila diperhatikan, setiap *nucleotide* yang terdapat dalam kolom ketiga bersebelahan dengan gap dalam kolom keempat, dan setiap *nucleotide* yang terdapat dalam kolom keempat bersebelahan dengan gap dalam kolom ketiga. Maka dapat dibuat solusi baru yaitu

```
- G G A - A
T G C A C A
- G G A C A
T T C A C A
```

yang memiliki skor 7.

Cara kedua didasari dari mungkinnya tercipta suatu kolom yang hanya berisi gap. Kemungkinan tersebut bisa terjadi karena pada dasarnya masing-masing dari *firefly algorithm*, *cuckoo search*, dan *flower pollination algorithm*

adalah algoritma yang bergerak secara acak. Sehingga cara kedua untuk membuat solusi baru adalah dengan menghapus gap yang terdapat dalam solusi sebelumnya.

4.4.3 Perbandingan Hasil Algoritma

Perbandingan hasil skor dari *multiple sequence alignment* dengan *firefly algorithm*, *cuckoo search*, dan *flower pollination algorithm* ditampilkan dalam Tabel 4.7, Tabel 4.8, dan Tabel 4.9. Dalam tabel-tabel tersebut, FA adalah *firefly algorithm* dengan solusi awal acak, FAm adalah *firefly algorithm* dengan solusi awal dari *modified* Needleman-Wunsch *alignment*, CS adalah *cuckoo search* dengan solusi awal acak, CSm adalah *cuckoo search* dengan solusi awal dari *modified* Needleman-Wunsch *alignment*, FPA adalah *flower pollination algorithm* dengan solusi awal acak, FPAm adalah *flower pollination algorithm* dengan solusi awal dari *modified* Needleman-Wunsch *alignment*.

Dari 150 kasus yang ada dalam Tabel 4.7, Tabel 4.8, dan Tabel 4.9, skor tertinggi dicapai oleh FA pada 22 kasus, FAm pada 130 kasus, CS pada 6 kasus, CSm pada 93 kasus, FPA pada 6 kasus, dan FPAm pada 74 kasus. Sehingga secara keseluruhan, *firefly algorithm* dengan solusi awal dari *modified* Needleman-Wunsch *alignment* adalah algoritma yang memiliki skor terbaik, namun memiliki waktu komputasi yang lebih lama dari yang lainnya.

Tabel 4.7 Perbandingan Hasil dari *Firefly Algorithm*, *Cuckoo Search*, dan *Flower Pollination Algorithm* untuk *Multiple Sequence Alignment* untuk 3 *Sequence*

l	FA		FAm		CS		CSm		FPA		FPAm	
	Skor	Waktu	Skor	Waktu	Skor	Waktu	Skor	Waktu	Skor	Waktu	Skor	Waktu
10	**15**	0.749	**15**	0.78	**15**	0.209	**15**	0.253	**15**	0.196	**15**	0.221
20	**13**	0.842	**13**	0.874	12	0.16	12	0.159	**13**	0.146	12	0.117
30	**41**	0.78	**41**	0.951	**41**	0.16	**41**	0.182	**41**	0.099	**41**	0.121
40	**50**	1.061	**50**	1.077	**50**	0.21	**50**	0.241	49	0.135	**50**	0.146
50	**64**	0.983	**64**	1.014	**64**	0.187	**64**	0.205	**64**	0.104	**64**	0.146
60	**77**	1.248	**77**	1.294	61	0.22	**77**	0.233	74	0.11	**77**	0.138
70	**82**	1.56	81	1.514	70	0.25	80	0.276	80	0.144	80	0.202
80	114	1.716	**116**	1.825	87	0.239	**116**	0.323	87	0.15	**116**	0.212
90	83	1.731	**91**	1.716	74	0.286	**91**	0.31	75	0.157	**91**	0.197
100	121	1.903	120	2.137	119	0.301	**122**	0.375	117	0.161	**122**	0.347
110	122	1.888	**134**	2.122	108	0.277	132	0.411	108	0.138	132	0.318
120	**149**	2.106	**149**	2.246	140	0.338	148	0.408	142	0.195	147	0.27
130	**167**	2.246	164	2.449	148	0.348	164	0.481	160	0.211	164	0.278
140	184	2.309	**185**	2.512	181	0.407	**185**	0.45	166	0.194	**185**	0.332
150	189	2.325	**198**	2.792	179	0.398	**198**	0.516	184	0.21	**198**	0.32
160	**202**	2.745	**202**	3.027	182	0.443	**202**	0.563	185	0.23	**202**	0.386
170	160	2.668	**170**	3.104	160	0.492	**170**	0.598	160	0.235	**170**	0.419
180	170	2.636	179	3.214	161	0.472	**186**	0.574	158	0.242	176	0.36
190	208	2.98	**216**	3.494	205	0.469	**216**	0.632	201	0.26	**216**	0.347
200	241	3.12	**248**	3.573	216	0.512	247	0.631	209	0.285	247	0.401
210	219	3.229	**221**	3.65	206	0.503	**221**	0.68	204	0.276	220	0.407
220	251	3.323	**263**	3.853	243	0.545	260	0.817	235	0.294	261	0.418
230	250	3.572	**256**	4.15	204	0.575	**256**	0.778	204	0.31	**256**	0.434
240	251	3.619	**301**	4.399	225	0.576	**301**	0.89	236	0.317	**301**	0.521
250	238	3.9	**284**	4.602	220	0.637	**284**	0.875	221	0.342	**284**	0.487
260	267	3.869	**297**	4.477	255	0.629	**297**	0.731	251	0.333	**297**	0.492
270	336	4.025	**342**	4.992	313	0.643	**342**	0.895	312	0.35	**342**	0.518
280	**324**	3.962	316	5.101	295	0.649	316	0.949	295	0.346	316	0.611
290	324	4.009	**338**	5.211	312	0.664	**338**	1.064	320	0.364	337	0.618
300	342	4.197	**364**	5.319	332	0.703	**364**	1.032	325	0.388	**364**	0.659
310	348	4.383	**368**	5.648	329	0.702	**368**	1.186	327	0.377	366	0.653
320	307	4.446	**370**	5.818	283	0.72	**370**	1.114	312	0.443	**370**	0.603
330	366	4.54	**405**	6.474	299	0.794	**405**	1.302	363	0.429	**405**	0.909
340	344	4.43	**365**	6.412	353	0.839	363	1.287	331	0.424	363	0.875
350	370	4.899	**402**	6.661	351	0.945	**402**	1.3	346	0.425	**402**	1.066
360	343	5.179	418	7.005	340	0.848	**419**	1.347	340	0.458	418	0.937
370	396	5.117	**454**	6.957	385	0.902	**454**	1.448	380	0.459	**454**	0.962
380	433	5.335	461	7.129	432	0.85	**463**	1.366	418	0.47	461	0.729
390	439	5.678	**449**	7.208	410	0.887	**449**	1.344	414	0.495	448	1.135
400	432	5.538	**472**	7.628	409	0.957	**472**	1.389	389	0.476	**472**	1.017
410	404	5.788	463	7.738	396	0.924	**464**	1.534	396	0.513	463	1.006
420	415	5.788	**483**	8.034	389	0.921	482	1.51	402	0.569	480	1.157
430	412	5.943	**471**	8.33	410	0.947	**471**	1.483	403	0.533	**471**	0.886
440	476	6.209	**490**	8.409	454	1.092	**490**	1.567	438	0.543	**490**	1.12
450	505	6.349	**535**	8.767	479	0.995	**535**	1.407	476	0.549	**535**	1.438
460	507	6.381	**578**	8.876	480	1.032	**578**	2.066	501	0.575	**578**	1.033
470	470	6.411	**560**	9.532	449	1.139	**560**	1.877	444	0.625	**560**	1.422
480	481	6.552	**566**	10.015	457	1.089	565	2.231	463	0.593	565	1.196
490	471	7.005	**524**	9.781	448	1.139	521	1.829	399	0.611	521	1.352
500	581	7.222	**624**	9.781	529	1.036	**624**	2.006	505	0.532	**624**	1.273

Tabel 4.8 Perbandingan Hasil dari *Firefly Algorithm*, *Cuckoo Search*, dan *Flower Pollination Algorithm* untuk *Multiple Sequence Alignment* untuk 4 *Sequence*

l	FA		FAm		CS		CSm		FPA		FPAm	
	Skor	Waktu	Skor	Waktu	Skor	Waktu	Skor	Waktu	Skor	Waktu	Skor	Waktu
10	**20**	0.765	**20**	0.812	**20**	0.132	**20**	0.127	**20**	0.067	**20**	0.069
20	**27**	1.232	26	1.232	26	0.198	26	0.198	**27**	0.098	26	0.11
30	64	1.201	**65**	1.466	64	0.207	**65**	0.252	64	0.114	63	0.143
40	**78**	1.623	77	1.857	75	0.277	77	0.314	75	0.143	77	0.187
50	102	1.919	**104**	1.934	88	0.31	**104**	0.329	82	0.17	**104**	0.192
60	124	1.996	**130**	2.262	105	0.355	**130**	0.376	114	0.18	**130**	0.222
70	144	2.465	**152**	2.73	130	0.389	**152**	0.461	132	0.22	**152**	0.285
80	**226**	2.699	225	3.058	194	0.392	225	0.531	191	0.211	223	0.313
90	**157**	2.98	155	3.463	139	0.505	155	0.595	122	0.283	155	0.377
100	223	3.088	**234**	3.495	222	0.485	232	0.583	216	0.271	232	0.395
110	245	3.027	263	3.572	245	0.534	**266**	0.67	213	0.221	263	0.391
120	266	3.307	**282**	3.978	254	0.544	281	0.698	256	0.308	**282**	0.496
130	**295**	3.604	288	4.196	269	0.569	286	0.766	273	0.324	286	0.506
140	316	3.744	**326**	4.571	304	0.619	**326**	0.828	306	0.335	**326**	0.47
150	313	4.056	**349**	5.242	288	0.625	**349**	0.957	292	0.336	**349**	0.595
160	353	4.711	**383**	5.694	335	0.745	**383**	0.964	330	0.407	**383**	0.669
170	337	4.618	**342**	5.85	328	0.762	338	1.02	326	0.41	338	0.658
180	319	4.383	**345**	5.99	311	0.713	**345**	1.085	320	0.394	**345**	0.613
190	**441**	5.101	433	6.412	404	0.796	430	1.251	403	0.447	430	0.789
200	385	5.445	**448**	6.973	409	0.895	**448**	1.185	319	0.481	**448**	0.779
210	387	5.241	**434**	7.02	379	0.885	**434**	1.234	392	0.496	**434**	0.886
220	480	5.741	**499**	7.348	467	0.928	495	1.454	467	0.519	495	0.793
230	349	6.178	**444**	8.158	355	0.994	**444**	1.345	350	0.561	**444**	0.936
240	435	6.084	**528**	8.69	426	1.008	**528**	1.571	420	0.561	**528**	0.976
250	427	6.63	**531**	9.032	397	1.192	530	1.767	400	0.61	530	0.982
260	515	7.082	**578**	9.579	431	1.115	577	1.697	426	0.629	577	1.461
270	594	6.942	**615**	10.171	571	1.096	**615**	2.001	586	0.614	**615**	1.261
280	579	6.88	**616**	10.389	560	1.125	614	2.02	568	0.635	**616**	1.438
290	577	7.691	**608**	11.466	518	1.218	**608**	2.383	519	0.678	**608**	1.482
300	595	7.987	**668**	12.044	561	1.241	**668**	1.962	552	0.701	**668**	1.638
310	653	7.722	**716**	11.637	651	1.258	708	2.147	631	0.707	708	1.392
320	546	8.315	**693**	12.714	520	1.419	**693**	2.141	575	0.751	**693**	1.563
330	693	9.219	**774**	12.699	659	1.425	**774**	2.293	647	0.808	770	1.641
340	608	9.158	**700**	13.4	570	1.424	**700**	2.322	572	0.793	**700**	1.743
350	637	9.734	**743**	14.648	553	1.494	740	2.721	578	0.849	738	1.604
360	623	9.578	**734**	14.524	617	1.557	**734**	2.78	581	0.858	**734**	1.98
370	844	10.047	**857**	14.321	770	1.539	**857**	2.963	749	0.868	**857**	1.915
380	743	9.797	**807**	15.382	691	1.575	806	2.91	722	0.934	806	2.161
390	796	10.186	**827**	15.631	766	1.627	825	2.707	724	0.925	825	2.637
400	804	10.094	**866**	15.615	784	1.608	862	2.89	774	0.922	862	2.312
410	810	10.436	**907**	16.724	795	1.628	903	3.603	788	0.927	903	2.694
420	790	10.67	**947**	16.926	789	1.817	**947**	4.4	781	0.988	**947**	2.586
430	813	11.685	**923**	16.864	828	1.856	922	3.417	779	1.044	921	2.297
440	810	11.887	**948**	18.19	795	1.906	**948**	3.633	820	1.087	946	3.166
450	942	12.387	**1013**	18.314	897	1.97	1011	3.769	929	1.126	1008	2.201
460	915	12.604	**1101**	18.658	900	2.008	**1101**	3.824	947	1.302	**1101**	2.914
470	934	12.496	**1038**	19.406	911	2.072	**1038**	4.448	905	1.225	1036	2.994
480	920	13.681	**1059**	20.795	893	2.071	**1059**	3.867	863	1.181	**1059**	3.103
490	899	14.227	**1060**	20.888	826	2.22	1056	3.82	837	1.268	1054	2.922
500	1038	14.274	**1139**	19.329	1010	2.177	**1139**	5.543	988	1.249	1134	3.482

Tabel 4.9 Perbandingan Hasil dari *Firefly Algorithm*, *Cuckoo Search*, dan *Flower Pollination Algorithm* untuk *Multiple Sequence Alignment* untuk 5 *Sequence*

l	FA		FAm		CS		CSm		FPA		FPAm	
	Skor	Waktu	Skor	Waktu	Skor	Waktu	Skor	Waktu	Skor	Waktu	Skor	Waktu
10	**33**	1.061	31	1.107	31	0.171	31	0.171	27	0.086	31	0.091
20	19	1.794	23	1.841	**24**	0.269	23	0.308	11	0.148	23	0.174
30	**122**	1.794	**122**	2.075	119	0.306	**122**	0.326	118	0.154	**122**	0.193
40	**108**	2.527	106	2.855	91	0.432	**108**	0.488	101	0.231	106	0.284
50	142	2.715	**163**	3.213	135	0.483	**163**	0.499	138	0.251	**163**	0.347
60	**216**	3.026	198	3.386	184	0.497	198	0.574	184	0.271	198	0.34
70	196	3.463	**228**	4.383	187	0.577	**228**	0.73	192	0.315	**228**	0.43
80	326	4.01	**337**	4.68	297	0.615	**337**	0.713	302	0.335	337	0.518
90	251	4.57	**294**	5.601	219	0.727	**294**	0.882	199	0.415	**294**	0.54
100	321	4.727	**341**	5.522	310	0.771	**341**	0.919	303	0.446	**341**	0.546
110	374	4.54	393	6.1	332	0.645	393	0.983	332	0.343	**394**	0.591
120	398	5.039	**445**	6.536	400	0.843	444	1.09	401	0.478	444	0.724
130	428	5.553	**437**	6.88	398	0.882	435	1.12	389	0.506	435	0.77
140	493	5.85	**490**	7.612	482	0.947	489	1.194	483	0.529	**490**	0.853
150	508	6.63	534	8.83	450	0.965	**540**	1.442	491	0.616	533	0.931
160	561	7.364	**600**	9.656	524	1.178	**600**	1.672	538	0.663	**600**	0.986
170	543	7.269	**546**	10.624	539	1.174	**546**	1.758	539	0.652	**546**	1.032
180	540	7.863	**594**	11.341	528	1.241	**594**	1.894	518	0.707	**594**	1.279
190	677	7.893	**713**	11.919	704	1.331	**713**	1.997	706	0.767	**713**	1.295
200	528	8.783	**723**	12.464	511	1.373	**723**	2.115	497	0.789	**723**	1.333
210	643	8.986	**681**	12.309	640	1.458	**681**	1.985	651	0.884	**681**	1.654
220	690	9.765	**802**	14.414	735	1.615	799	2.609	699	0.916	799	1.729
230	667	10.561	**761**	14.57	590	1.61	760	2.558	659	0.966	760	1.531
240	752	10.281	**858**	15.132	668	1.572	856	2.777	713	0.922	856	1.943
250	704	11.294	866	16.864	694	1.793	**867**	2.944	735	1.064	**867**	2.031
260	733	12.122	**885**	16.365	708	1.826	**885**	3.058	779	1.181	885	1.876
270	938	11.84	**969**	17.503	922	1.835	**969**	3.032	933	1.121	**969**	2.266
280	920	12.106	**995**	18.299	899	1.924	994	3.174	886	1.115	**995**	2.116
290	836	13.525	938	19.188	804	2.068	**939**	3.641	804	1.223	935	3.018
300	901	14.508	**1022**	20.67	884	2.273	1017	3.358	837	1.236	1017	3.01
310	1022	13.135	**1104**	19.468	1015	2.149	1091	4.178	995	1.246	1091	2.673
320	906	14.602	**1101**	22.511	869	2.289	1097	4.752	861	1.311	1093	3.31
330	1014	16.77	**1168**	21.684	906	2.525	1167	4.438	963	1.529	1162	2.913
340	918	16.317	1082	24.539	840	2.499	1082	5.113	823	1.455	1082	2.984
350	999	17.27	**1169**	25.506	952	2.857	**1169**	5.376	910	1.54	**1169**	3.201
360	1050	17.737	**1194**	25.428	1000	2.744	1193	4.941	922	1.596	1190	3.324
370	1195	16.863	**1393**	23.79	1191	2.755	**1393**	4.814	1198	1.688	**1393**	4.165
380	1157	17.488	**1267**	26.021	1155	2.817	**1267**	5.966	1126	1.623	1266	4.097
390	1024	19.079	**1251**	27.955	1011	2.935	1248	5.742	1012	1.786	1247	3.994
400	1330	18.767	**1377**	28.018	1333	3.138	1366	6.55	1310	1.773	1368	4.967
410	1291	19.859	**1397**	27.285	1235	3.037	1389	6.565	1217	1.764	1387	4.144
420	1320	19.952	**1522**	28.969	1273	3.128	1511	6.052	1303	1.893	1513	4.989
430	1173	21.934	**1422**	31.574	1150	3.491	**1422**	5.813	1225	2.14	1415	5.2
440	1302	21.902	1477	31.622	1260	3.49	**1479**	6.128	1258	2.062	1472	5.208
450	1478	22.105	**1579**	34.061	1417	3.667	1570	6.339	1473	2.124	1571	4.88
460	1389	22.355	**1671**	36.736	1428	3.653	1664	7.363	1362	2.155	1664	5.607
470	1450	21.949	**1573**	35.85	1436	3.553	1566	9.049	1433	2.092	1563	4.374
480	1500	24.243	**1688**	38.213	1422	3.862	1676	8.404	1435	2.317	1673	5.075
490	1359	24.929	**1685**	37.718	1359	4.16	1676	7.703	1396	2.462	1673	5.625
500	1698	26.255	**1842**	37.705	1652	4.135	**1842**	6.851	1655	2.457	1838	6.041

BAB V KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasar pada pembahasan yang telah dipaparkan, dapat diambil beberapa kesimpulan yaitu:

1. Modifikasi yang dilakukan pada *firefly algorithm, cuckoo search,* dan *flower pollination algorithm* dapat digunakan untuk menyelesaikan masalah optimasi fungsi dengan hasil dan waktu komputasi yang baik.
2. *Modified* Needleman-Wunsch *alignment* yang telah dipaparkan adalah metode yang cukup baik. *Modified* Needleman-Wunsch *alignment* memiliki skor yang lebih baik dari *star alignment*. Hasil skor yang diperoleh untuk pengujian dengan tiga *sequence* tidak jauh berbeda dari Needleman-Wunsch *alignment* yang asli.
3. *Firefly algorithm, cuckoo search,* dan *flower pollination algorithm* untuk *multiple sequence alignment* menggunakan langkah-langkah yang hampir sama dengan yang digunakan untuk optimasi fungsi. Perbedaannya adalah digunakannya dua cara tersendiri untuk membentuk solusi baru. Solusi awal untuk algoritma-algoritma tersebut didapatkan dari pembentukan acak dan metode baru Needleman-Wunsch *alignment*. Tampak bahwa tiga algoritma tersebut dapat menghasilkan solusi-solusi baru yang lebih baik. Secara keseluruhan, *firefly algorithm* adalah algoritma yang memiliki skor lebih baik, namun memiliki waktu komputasi yang lebih lama.

5.2 Saran

Saran yang diberikan oleh penulis untuk penelitian selanjutnya antara lain adalah:

1. Menggunakan metode baru Needleman-Wunsch *alignment* untuk *sequence* protein.
2. Mengembangkan metode baru Needleman-Wunsch *alignment* untuk model gap *affine*.

3. Dalam penelitian ini, peneliti hanya mampu untuk membuat dua cara untuk membentuk solusi *multiple sequence alignment* baru. Akan lebih bagus apabila dapat dibentuk beberapa cara baru yang lain, terutama apabila dapat disesuaikan dengan karakteristik dari masing-masing algoritma yang digunakan.
4. Menggunakan algoritma-algoritma yang telah dipaparkan untuk mengerjakan *multiple sequence alignment* dengan *sequence* protein.
5. Mengembangkan algoritma-algoritma yang telah dipaparkan untuk mengerjakan *multiple sequence alignment* dengan model gap *affine*.

DAFTAR PUSTAKA

Abdelaziz, A. Y., Ali, E. S., dan Elazim, S. M. (2016), "Combined Economic and Emission Dispatch Solution Using Flower Pollination Algorithm", *International Journal of Electrical Power& Systems*, Vol. 80, hal 264-274.

Apostolopoulos, T. dan Vlachos, A. (2011), "Application of the Firefly Algorithm for Solving the Economic Emissions Load Dispatch Problem". *Int J Combin*.

Barton, G. J. dan Sternberg, M. J. E. (1987), "A Strategy for The Rapid Multiple Alignment of Protein Sequences", *J Mol Biol*, Vol. 198, hal. 327-337.

Chandrasekaran, K., Simon, S. P. (2012), "Multi-objective scheduling problem: hybrid appraoch using fuzzy assisted cuckoo search algorithm", *Swarm Evol Comput*, Vol. 5(1), hal. 1-6.

Dahi, Z. A. E. M., Mezioud, C., dan Draa, A. (2016), "On the Efficiency of The Binary Flower Pollination Algorithm: Application on the Antenna Positioning Problem", *Applied Soft Computing*.

Devereux, J., Haeberli, P., dan Smithies, O. (1984), "Acomprehensive Set of Sequence Analysis Programs for The Vax", *Nucleic Acids Res*, Vol. 12, hal. 387-395.

Fister, I. J. R., Fister, I., Brest, J., Yang, X. S. (2012), "Memetic Firefly Algorithm for Combinatorial Optimization", *Bioinspired optimization methods and their applications* (BIOMA2012), hal. 75-86.

Higgins, D. G., Bleasby, A. J., dan Fuchs, J. A. (1992), "CLUSTAL V: Improved Software for Multiple Sequence Alignment", *Bioinformatics*, Vol. 8, hal. 189-191.

Isaev, A. (2006), *Introduction to Mathematical Methods in Bioinformatics*, Springer, Berlin.

Jati, G. K. Dan Suyanto, S. (2011) "Evolutionary Discrete Firefly Algorithm for Traveling Salesman Problem", *Lecture Notes in Artificial Intelligence*, hal. 393-403.

Layeb, A. (2011), "A Novel Quantum-Inspired Cuckoo Search for Knapsack Problems", *Int J Bioinspired Comput*, Vol. 3(5), hal. 297-305.

Lee, Z. J., Su, S. F., Chuang, C. C., dan Liu, K. H. (2008), "Genetic Algorithm with Ant Colony Optimization (GA-ACO) for Multiple Sequence Alignment", *Applied Soft Computing*, Vol. Vol. 8, hal. 55-78.

Nabil, E. (2016), "A Modified Flower Pollination Algorithm for Global Optimization", *Expert Systems With Applications*, Vol. 57, hal. 192-203.

Naznin, F., Sarker, R., Essam, D. (2011), "Vertical Decomposition with Genetic Algorithm for Multiple Sequence Alignment", *BMC Bioinformatics*, Vol. 12.

Needleman, S. B. dan Wunsch C. D. (1970), "A General Method Applicable to The Search of Similarities in The Amino Acid Sequence of Two Proteins", *J Mol Biol*, Vol. 48, hal. 443-453.

Notredame, C. dan Higgins, D. G. (1996), "SAGA: Sequence Alignment by Genetic Algorithm", *Nucleic Acids Res*, Vol. 24, hal. 1515-1524.

Notredame, C., Higgins, D. G., dan Heringa, J. (1987), "T-Coffee: A Novel Method for Fast and Accurate Multiple Sequence Alignment", *J Mol Biol*, Vol. 4, hal. 406-425.

Rampriya, B., Mahadevan, K., dan Kannan S. (2010), "Unit Commitment in Deregulated Power System Using Lagrangian Firefly Algorithm", *Proceedings of IEEE International Conference on Communication Control and Computing Technologies*, hal. 389-393.

Shaffer, C. A. (2013), *Data Structures and Algorithm Analysis*, Dover Publications, Blacksburg.

Shahab, M. L., Daryono, B. U., dan Irawan, M. I. (2016), "Decomposing and Solving Capacitated Vehicle Routing Problem (CVRP) using Two-Step Genetic Algorithm (TSGA)", *Journal of Theoritical and Applied Information Technology*, Vol. 87(3), hal. 461-468.

Shen, S. N. dan Tuszynski, J. A. (2008), *Theory and Mathematical Methods for Bioinformatics*, Springer, Berlin.

Shyu, C., Sheneman, L., dan Foster, J. A. (2004), "Multiple Sequence Alignment with Evolutionary Computation", *Genetic Programming and Evolvable Machines*, Vol. 5, hal. 121-144.

Stoye, J., Perrey, S. W., dan Dress, A. W. M. (1997), "Improving The Divide and Concuer Approach to Sum of Pairs Multiple Sequence Alignment", *App Maths Letters*, Vol. 10, hal. 67-73.

Taylor, W. (1988), "A Flexible Method to Align Large Numbers of Biological Sequences", *J Mol Biol*, Vol. 28, hal. 61-69.

Thompson, J. D., Gibson, T. J., Plewniak, F., Jeanmougin, F., dan Higgins, D. G. (1997), "The CLUSTAL_X Windows Interface: Flexible Strategies for Multiple Sequence Alignment Aided by Quality Analysis Tools", *Nucleic Acids Res*, Vol. 24, hal. 4876-4882.

Thompson, J. D., Higgins, D. G., dan Gibson, T. J. (1994), "CLUSTAL_W: Improving The Sensitivity of Progressive Multiple Sequence Alignment Through Sequence Weighting, Position-Specific Gap Penalties and Weight Matrix Choice", *Nucleic Acids Res*, Vol. 22, hal. 4673-4680.

Thomsen, R., Fogel, G. B., dan Krink, T. (2002), "A Clustal Alignment Improver Using Evolutionary Algorithms", *Congress on Evolutionary Computation*, Vol. 1, hal. 121-126.

Valian, E., Mohanna, S., dan Tavakoli, S. (2011), "Improved Cuckoo Search Algorithm for Feedforward Neural Network Training", *Int J Artif Intell Appl*, Vol. 2(3), hal. 36-43.

Wang, R., Zhou, Y., Qiao, S., dan Huang, K. (2016), "Flower Pollination Algorithm with Bee Pollinator for Cluster Analysis", *Information Processing Letters*, Vol. 116(1), hal. 1-14.

Yang, X. S. (2013), "Multiobjective Firefly Algorithm for Continuous Optimization", *Eng Comput*, Vol. 29(2), hal. 175-184.

Yang, X. S. (2014), *Nature-Inspired Optimization Algorithms*, Elsevier, London.

LAMPIRAN

Lampiran 1 *Source Code* dari Main.java dalam *Package* FunctionOptimization

```
package FunctionOptimization;

public class Main {

    public static void main(String[] args) {
        int solution_size[] = {10, 25, 50, 100, 250, 500};
        double tollerance[] = {1.0E-10, 1.0E-100};
        int population_size = 10;
        double domain_range = 100;
        Data.set_population_size(population_size);
        Data.set_domain_range(domain_range);

        for (int i = 0; i < solution_size.length; i++){
            Data.set_solution_size(solution_size[i]);
            for (int j = 0; j < 2; j++){
                long startTime = System.currentTimeMillis();
                Population population = new
Population(Data.get_population_size(), true);
                while (population.get_solution(0).get_fitness() >
tollerance[j]){
                    population = FireflyAlgorithm.iteration(population);
                    //population =
FireflyAlgorithmBefore.iteration(population);
                    //population = CuckooSearch.iteration(population);
                    //population =
CuckooSearchBefore.iteration(population);
                    //population =
FlowerPollinationAlgorithm.iteration(population);
                    //population =
FlowerPollinationAlgorithmBefore.iteration(population);
                }
                long endTime = System.currentTimeMillis() - startTime;
                System.out.println((double)(endTime)/1000);
            }
        }
    }
}
```

Lampiran 2 *Source Code* dari Data.java dalam *Package* FunctionOptimization

```
package FunctionOptimization;

public class Data {
    private static int population_size;
    private static int solution_size;
    private static double domain_range;

    public static void set_population_size(int value){
        population_size = value;
    }

    public static int get_population_size(){
        return population_size;
    }

    public static void set_solution_size(int value){
        solution_size = value;
    }

    public static int get_solution_size(){
        return solution_size;
    }

    public static void set_domain_range(double value){
        domain_range = value;
    }

    public static double get_domain_range(){
        return domain_range;
    }
}
```

Lampiran 3 *Source Code* dari Solution.java dalam *Package* FunctionOptimization

```
package FunctionOptimization;

public class Solution {
    private double solution[];
    private double fitness;

    public Solution(){
        solution = new double[Data.get_solution_size()];
        for (int i = 0; i < Data.get_solution_size(); i++){
            solution[i]   =   Math.random()*2*Data.get_domain_range()   -
Data.get_domain_range();
        }
    }

    public void set_element(int index, double value){
        solution[index] = value;
        fitness = 0;
    }

    public double get_element(int index){
        return solution[index];
    }

    public double get_fitness(){
        if (fitness == 0){
            for (int i = 0; i < Data.get_solution_size(); i++){
                fitness = fitness + (solution[i]-50)*(solution[i]-50);
            }
        }
        return fitness;
    }

    public void print_solution(){
        for (int i = 0; i < Data.get_solution_size(); i++){
            System.out.print(solution[i] + " ");
        }
        System.out.println("");
    }

    public void print_fitness(){
        System.out.println(get_fitness());
    }
}
```

Lampiran 4 *Source Code* dari Population.java dalam *Package* FunctionOptimization

```
package FunctionOptimization;

public class Population {
    Solution solution[];
    int population_size;

    public Population(int n, boolean b){
        solution = new Solution[n];
        population_size = n;
        if (b == true){
            for (int i = 0; i < n; i++){
                solution[i] = new Solution();
            }
            sort();
        }
    }
    public void save_solution(int index, Solution value){
        solution[index] = value;
    }

    public void sort(){
        for(int i = 1; i < population_size; i++){
            for(int  j  =  i;  j  >  0  &&  solution[j].get_fitness()  <
solution[j-1].get_fitness(); j--){
                Solution dummy = solution[j-1];
                solution[j-1] = solution[j];
                solution[j] = dummy;
            }
        }
    }
    public Solution get_solution(int index) {
        return solution[index];
    }
    public Solution getFittest() {
        return solution[0];
    }
    public int get_population_size() {
        return population_size;
    }
    public void print_all_fitness(){
        for (int i = 0; i < population_size; i++){
            solution[i].print_fitness();
        }
    }
}
```

Lampiran 5 *Source Code* dari FireflyAlgorithm.java dalam *Package* FunctionOptimization

```
package FunctionOptimization;

public class FireflyAlgorithm {

    public static Population iteration(Population population){
        Population new_population1 = population;
        for (int i = 0; i < population.get_population_size(); i++){
            for (int j = 0; j < population.get_population_size(); j++){
                Solution new_solution =
move_to(population.get_solution(i), population.get_solution(j));
                if (new_solution.get_fitness() <
new_population1.get_solution(i).get_fitness()){
                    new_population1.save_solution(i, new_solution);
                }
            }
        }
        return new_population1;
    }

    public static Solution move_to(Solution a, Solution b){
        Solution new_solution = new Solution();
        for (int i = 0; i < Data.get_solution_size(); i++){
            new_solution.set_element(i, a.get_element(i) +
Math.random()*2*(Math.random()-0.5)*(a.get_element(i)-50) +
Math.random()*(b.get_element(i)-a.get_element(i)));
        }
        return new_solution;
    }
}
```

Lampiran 6 *Source Code* dari CuckooSearch.java dalam *Package* FunctionOptimization

```
package FunctionOptimization;

public class CuckooSearch {
    private static double pa = 0.5;

    public static Population iteration(Population population){
        Population new_population1 = new
Population(Data.get_population_size(), false);
        Population new_population2 = new
Population(2*Data.get_population_size(), false);
        for (int i = 0; i < population.get_population_size(); i++){
            new_population2.save_solution(i, population.get_solution(i));

new_population2.save_solution(population.get_population_size()+i,
global_random_walk(population.get_solution(i)));
        }
        new_population2.sort();

        int m = 0;
        new_population1.save_solution(0,
new_population2.get_solution(0));
        for (int i = 1; i < new_population2.get_population_size() && m <
population.get_population_size()-1; i++){
            if (new_population2.get_solution(i).get_fitness() !=
new_population1.get_solution(m).get_fitness()){
                m++;
                new_population1.save_solution(m,
new_population2.get_solution(i));
            }
        }

        if ((m+1) != population.get_population_size()){
            for (int i = m+1; i < population.get_population_size(); i++){
                Solution acak = new Solution();
                new_population1.save_solution(i, acak);
            }
        }

        for (int i = 0; i < population.get_population_size(); i++){
            if (Math.random() < pa){
                Solution new_solution =
local_random_walk(new_population1.get_solution(i),
new_population1.get_solution((int)(Math.random()*population.get_populatio
n_size())),
new_population1.get_solution((int)(Math.random()*population.get_populatio
n_size())));
                if (new_solution.get_fitness() <
new_population1.get_solution(i).get_fitness()){
                    new_population1.save_solution(i, new_solution);
                }
```

```
            }
        }

        return new_population1;
    }

    public static Solution local_random_walk(Solution solution1, Solution
solution2, Solution solution3){
        Solution new_solution = new Solution();
        for (int i = 0; i < Data.get_solution_size(); i++){
            new_solution.set_element(i, solution1.get_element(i) +
Math.random()*(solution2.get_element(i)-solution3.get_element(i)));
        }
        return new_solution;
    }

    public static Solution global_random_walk(Solution solution){
        Solution new_solution = new Solution();
        for (int i = 0; i < Data.get_solution_size(); i++){
            new_solution.set_element(i, solution.get_element(i) +
(((solution.get_element(i)-50)*Math.random()*2*(Math.random()-0.5))));
        }
        return new_solution;
    }
}
```

Lampiran 7 *Source Code* dari FlowerPollinationAlgorithm.java dalam *Package* FunctionOptimization

```
package FunctionOptimization;

public class FlowerPollinationAlgorithm {
    private static double p = 0.1;

    public static Population iteration(Population population){
        Population new_population1 = new
Population(population.get_population_size(), false);
        for (int i = 0; i < population.get_population_size(); i++){
            if (Math.random() < p){
                new_population1.save_solution(i,
global_pollination(population.get_solution(i), population.getFittest()));
            }
            else{
                new_population1.save_solution(i,
local_pollination(population.get_solution(i),
population.get_solution((int)(Math.random()*population.get_population_siz
e())),
population.get_solution((int)(Math.random()*population.get_population_siz
e())));
            }
        }

        Population new_population2 = new
Population(2*Data.get_population_size(), false);
        for (int i = 0; i < population.get_population_size(); i++){
            new_population2.save_solution(i, population.get_solution(i));

new_population2.save_solution(population.get_population_size()+i,
new_population1.get_solution(i));
        }
        new_population2.sort();

        int m = 0;
        new_population1.save_solution(0,
new_population2.get_solution(0));
        for (int i = 1; i < new_population2.get_population_size() && m <
population.get_population_size()-1; i++){
            if (new_population2.get_solution(i).get_fitness() !=
new_population1.get_solution(m).get_fitness()){
                m++;
                new_population1.save_solution(m,
new_population2.get_solution(i));
            }
        }

        if ((m+1) != population.get_population_size()){
            for (int i = m+1; i < population.get_population_size(); i++){
                Solution acak = new Solution();
                new_population1.save_solution(i, acak);
```

```
            }
        }

        return new_population1;
    }

    public static Solution global_pollination(Solution solution, Solution
best){
        Solution new_solution = new Solution();
        for (int i = 0; i < Data.get_solution_size(); i++){
            new_solution.set_element(i, solution.get_element(i) +
Math.random()*(best.get_element(i)-solution.get_element(i)));
        }
        return new_solution;
    }

    public static Solution local_pollination(Solution solution1, Solution
solution2, Solution solution3){
        Solution new_solution = new Solution();
        for (int i = 0; i < Data.get_solution_size(); i++){
            new_solution.set_element(i, solution1.get_element(i) +
Math.random()*2*(Math.random()-0.5)*(solution1.get_element(i)-50) +
Math.random()*(solution2.get_element(i)-solution3.get_element(i)));
        }
        return new_solution;
    }
}
```

Lampiran 8 *Source Code* dari SequenceGenerator.java dalam *Package* MultipleSequenceAlignment

```
package MultipleSequenceAlignment;

public class SequenceGenerator {

    public static String[] generate(int n, int l){
        String parent = generate_parent(l);

        // membentuk sequence dengan melakukan mutasi
        String sequence[] = new String[n];
        for (int i = 0; i < n; i++){
            sequence[i] = "";
            for (int j = 0; j < l; j++){
                double random = Math.random();
                // Type I - mutation
                if (random < 0.1){
                    double random1 = Math.random();
                    if (random1 < 0.25)
                        sequence[i] = sequence[i] + "A";
                    else if (random1 < 0.5)
                        sequence[i] = sequence[i] + "C";
                    else if (random1 < 0.75)
                        sequence[i] = sequence[i] + "G";
                    else
                        sequence[i] = sequence[i] + "T";
                }
                else if (random < 0.2){
                }
                else if (random < 0.3){
                    double random1 = Math.random();
                    if (random1 < 0.125)
                        sequence[i] = sequence[i] + "A" +
parent.charAt(j);
                    else if (random1 < 0.25)
                        sequence[i] = sequence[i] + "C" +
parent.charAt(j);
                    else if (random1 < 0.375)
                        sequence[i] = sequence[i] + "G" +
parent.charAt(j);
                    else if (random1 < 0.5)
                        sequence[i] = sequence[i] + "T" +
parent.charAt(j);
                    else if (random1 < 0.625)
                        sequence[i] = sequence[i] + parent.charAt(j) +
"A";
                    else if (random1 < 0.75)
                        sequence[i] = sequence[i] + parent.charAt(j) +
"C";
                    else if (random1 < 0.875)
                        sequence[i] = sequence[i] + parent.charAt(j) +
"G";
```

```
                    else
                        sequence[i] = sequence[i] + parent.charAt(j) +
"T";
                }
                // Type II - mutation
                else if (random < 0.4){
                    if (j < l-1){
                        sequence[i] = sequence[i] + parent.charAt(j+1) +
parent.charAt(j);
                        j++;
                    }
                    else {
                        sequence[i] = sequence[i] + parent.charAt(j);
                    }
                }
                // tidak terjadi mutasi
                else{
                    sequence[i] = sequence[i] + parent.charAt(j);
                }
            }
        }
        return sequence;
    }
    public static String generate_parent(int l){
        String parent = "";
        for (int i = 0; i < l; i++){
            double random = Math.random();
            if (random < 0.25){
                parent = parent + "A";
            }
            else if (random < 0.5){
                parent = parent + "C";
            }
            else if (random < 0.75){
                parent = parent + "G";
            }
            else {
                parent = parent + "T";
            }
        }
        return parent;
    }
}
```

Lampiran 9 *Source Code* dari Score.java dalam *Package* MultipleSequenceAlignment

```
package MultipleSequenceAlignment;

public class Score {
    private static int score[] = {1,0,0,-2};

    public static int count(String msa[]){
        int score = 0;
        for(int i = 0; i < msa.length; i++){
            for(int j = i+1; j < msa.length; j++){
                score = score + get_pair_sequence_score(msa[i], msa[j]);
            }
        }
        return score;
    }

    public static int get_pair_sequence_score(String a, String b){
        int score = 0;
        for(int i = 0; i < a.length(); i++){
            score = score + get_score(a.charAt(i), b.charAt(i));
        }
        return score;
    }

    public static int column_score(String a){
        int score = 0;
        for (int i = 0; i < a.length()-1; i++){
            for (int j = i+1; j < a.length(); j++){
                score = score + get_score(a.charAt(i), a.charAt(j));
            }
        }
        return score;
    }
    public static int get_score(char a, char b){
        if(a == '-' && b == '-')
            return score[2];
        else if(a == '-' || b == '-')
            return score[3];
        else if(a == b)
            return score[0];
        else
            return score[1];
    }
}
```

Lampiran 10 *Source Code* dari Star.java dalam *Package* MultipleSequenceAlignment

```
package MultipleSequenceAlignment;

public class Star {

    public static String[] alignment(String a[]){
        int score[][] =
NeedlemanWunsch.get_Needleman_Wunsch_pairs_score(a);

        int row_score[] = new int[a.length];
        for (int i = 0; i < a.length; i++){
            for (int j = 0; j < a.length; j++){
                row_score[i] = row_score[i] + score[i][j];
            }
        }

        int star = 0;
        int max = row_score[0];
        for (int i = 1; i < a.length; i++){
            if (row_score[i] > max){
                max = row_score[i];
                star = i;
            }
        }
        if (star != 0){
            String b = a[star];
            a[star] = a[0];
            a[0] = b;
        }
        String S[] = NeedlemanWunsch.sequence_alignment(a[0], a[1]);
        for (int i = 2; i < a.length; i++){
            String s[] = NeedlemanWunsch.sequence_alignment(a[0], a[i]);
            S = combine(S, s);
        }
        if (star != 0){
            String b = S[star];
            S[star] = S[0];
            S[0] = b;
            b = a[star];
            a[star] = a[0];
            a[0] = b;
        }

        return S;
    }


    public static String[] combine(String s1[], String s2[]){
        int i = 0;
        int j = 0;
        int n1 = s1.length;
```

```
        String s[] = new String[n1+1];
        for (int k = 0; k < n1+1; k++){
            s[k] = "";
        }
        while (i < s1[0].length() && j < s2[0].length()){
            if (s1[0].charAt(i) == s2[0].charAt(j)){
                for (int k = 0; k < n1; k++){
                    s[k] = s[k] + "" + s1[k].charAt(i);
                }
                s[n1] = s[n1] + "" + s2[1].charAt(j);
                i++;
                j++;
            }
            else if (s1[0].charAt(i) == '-'){
                for (int k = 0; k < n1; k++){
                    s[k] = s[k] + "" + s1[k].charAt(i);
                }
                s[n1] = s[n1] + "-";
                i++;
            }
            else if (s2[0].charAt(j) == '-'){
                for (int k = 0; k < n1; k++){
                    s[k] = s[k] + "-";
                }
                s[n1] = s[n1] + "" + s2[1].charAt(j);
                j++;
            }
        }
        while (i < s1[0].length()){
            for (int k = 0; k < n1; k++){
                s[k] = s[k] + "" + s1[k].charAt(i);
            }
            s[n1] = s[n1] + "-";
            i++;
        }
        while (j < s2[0].length()){
            for (int k = 0; k < n1; k++){
                s[k] = s[k] + "-";
            }
            s[n1] = s[n1] + "" + s2[1].charAt(j);
            j++;
        }
        return s;
    }
}
```

Lampiran 11 *Source Code* dari NeedlemanWunsch.java dalam *Package* MultipleSequenceAlignment

```
package MultipleSequenceAlignment;

public class NeedlemanWunsch {
    private static int score[] = {1,0,-2};
    private static int gap_with_gap_score = 0;

    public static String[] alignment(String a, String b, String c){
        int F[][][] = new int[a.length()+1][b.length()+1][c.length()+1];
        for (int i = 0; i < a.length()+1; i++){
            for (int j = 0; j < b.length()+1; j++){
                for (int k = 0; k < c.length()+1; k++){
                    if (i > 0 && j > 0 && k > 0){
                        int neighbor[] = new int[7];
                        neighbor[0] = F[i-1][j-1][k-1] +
get_score(a.charAt(i-1), b.charAt(j-1)) + get_score(a.charAt(i-1),
c.charAt(k-1)) + get_score(b.charAt(j-1), c.charAt(k-1));
                        neighbor[1] = F[i][j-1][k-1] +
get_score(b.charAt(j-1), c.charAt(k-1)) + 2*score[2];
                        neighbor[2] = F[i-1][j][k-1] +
get_score(a.charAt(i-1), c.charAt(k-1)) + 2*score[2];
                        neighbor[3] = F[i-1][j-1][k] +
get_score(a.charAt(i-1), b.charAt(j-1)) + 2*score[2];
                        neighbor[4] = F[i][j][k-1] + 2*score[2];
                        neighbor[5] = F[i][j-1][k] + 2*score[2];
                        neighbor[6] = F[i-1][j][k] + 2*score[2];
                        F[i][j][k] = get_max(neighbor);
                    }
                    else if (j > 0 && k > 0){
                        int neighbor[] = new int[3];
                        neighbor[0] = F[i][j-1][k-1] +
get_score(b.charAt(j-1), c.charAt(k-1)) + 2*score[2];
                        neighbor[1] = F[i][j][k-1] + 2*score[2];
                        neighbor[2] = F[i][j-1][k] + 2*score[2];
                        F[i][j][k] = get_max(neighbor);
                    }
                    else if (i > 0 && k > 0){
                        int neighbor[] = new int[3];
                        neighbor[0] = F[i-1][j][k-1] +
get_score(a.charAt(i-1), c.charAt(k-1)) + 2*score[2];
                        neighbor[1] = F[i][j][k-1] + 2*score[2];
                        neighbor[2] = F[i-1][j][k] + 2*score[2];
                        F[i][j][k] = get_max(neighbor);
                    }
                    else if (i > 0 && j > 0){
                        int neighbor[] = new int[3];
                        neighbor[0] = F[i-1][j-1][k] +
get_score(a.charAt(i-1), b.charAt(j-1)) + 2*score[2];
                        neighbor[1] = F[i][j-1][k] + 2*score[2];
                        neighbor[2] = F[i-1][j][k] + 2*score[2];
                        F[i][j][k] = get_max(neighbor);
```

```
                    }
                    else if (k > 0){
                        F[i][j][k] = F[i][j][k-1] + 2*score[2];
                    }
                    else if (j > 0){
                        F[i][j][k] = F[i][j-1][k] + 2*score[2];
                    }
                    else if (i > 0){
                        F[i][j][k] = F[i-1][j][k] + 2*score[2];
                    }
                    else {
                        F[i][j][k] = 0;
                    }
                }
            }
        }
        return back_track(F, a, b, c);
    }

    public static String[] back_track(int F[][], String a, String b){
        String c = "";
        int i = a.length();
        int j = b.length();

        while (i > 0 || j > 0){
            if (j == 0){
                c = a.charAt(i-1) + "-" + c;
                i--;
            }
            else if (i == 0){
                c = "-" + b.charAt(j-1) + c;
                j--;
            }
            else{
                if (F[i][j] == F[i-1][j-1] + get_score(a.charAt(i-1),
b.charAt(j-1))){
                    c = a.charAt(i-1) + "" + b.charAt(j-1) + c;
                    i--;
                    j--;
                }
                else if (F[i][j] == F[i-1][j] + score[2] && F[i][j] ==
F[i][j-1] + score[2]){
                    double r = Math.random();
                    if (r < 0.5){
                        c = a.charAt(i-1) + "-" + c;
                        i--;
                    }
                    else {
                        c = "-" + b.charAt(j-1) + c;
                        j--;
                    }
                }
                else if (F[i][j] == F[i-1][j] + score[2]){
                    c = a.charAt(i-1) + "-" + c;
                    i--;
                }
```

```
                else if (F[i][j] == F[i][j-1] + score[2]){
                    c = "-" + b.charAt(j-1) + c;
                    j--;
                }
            }
        }
        String d = "";
        String e = "";
        for (int index = 0; index < c.length()/2; index++){
            d = d + c.charAt(2*index);
            e = e + c.charAt(2*index+1);
        }
        String s[] = {d, e};
        return s;
    }

    public static String[] back_track(int F[][][], String a, String b,
String c){
        int i = a.length();
        int j = b.length();
        int k = c.length();
        String z = "";
        while (i > 0 && j > 0 && k > 0){
            int neighbor[] = new int[7];
            neighbor[0] = F[i-1][j-1][k-1] + get_score(a.charAt(i-1),
b.charAt(j-1)) + get_score(a.charAt(i-1), c.charAt(k-1)) +
get_score(b.charAt(j-1), c.charAt(k-1));
            neighbor[1] = F[i][j-1][k-1] + get_score(b.charAt(j-1),
c.charAt(k-1)) + 2*score[2];
            neighbor[2] = F[i-1][j][k-1] + get_score(a.charAt(i-1),
c.charAt(k-1)) + 2*score[2];
            neighbor[3] = F[i-1][j-1][k] + get_score(a.charAt(i-1),
b.charAt(j-1)) + 2*score[2];
            neighbor[4] = F[i][j][k-1] + 2*score[2];
            neighbor[5] = F[i][j-1][k] + 2*score[2];
            neighbor[6] = F[i-1][j][k] + 2*score[2];
            if (F[i][j][k] == neighbor[0]){
                z = a.charAt(i-1) + "" + b.charAt(j-1) + "" + c.charAt(k-
1) + z;
                i--;
                j--;
                k--;
            }
            else if (F[i][j][k] == neighbor[1]){
                z = "-" + b.charAt(j-1) + "" + c.charAt(k-1) + z;
                j--;
                k--;
            }
            else if (F[i][j][k] == neighbor[2]){
                z = a.charAt(i-1) + "-" + c.charAt(k-1) + z;
                i--;
                k--;
            }
            else if (F[i][j][k] == neighbor[3]){
                z = a.charAt(i-1) + "" + b.charAt(j-1) + "-" + z;
                i--;
```

```
                j--;
            }
            else if (F[i][j][k] == neighbor[4]){
                z = "-" + "-" + c.charAt(k-1) + z;
                k--;
            }
            else if (F[i][j][k] == neighbor[5]){
                z = "-" + b.charAt(j-1) + "-" + z;
                j--;
            }
            else if (F[i][j][k] == neighbor[6]){
                z = a.charAt(i-1) + "-" + "-" + z;
                i--;
            }
        }
        while (j > 0 && k > 0){
            int neighbor[] = new int[3];
            neighbor[0] = F[i][j-1][k-1] + get_score(b.charAt(j-1),
c.charAt(k-1)) + 2*score[2];
            neighbor[1] = F[i][j][k-1] + 2*score[2];
            neighbor[2] = F[i][j-1][k] + 2*score[2];
            if (F[i][j][k] == neighbor[0]){
                z = "-" + b.charAt(j-1) + "" + c.charAt(k-1) + z;
                j--;
                k--;
            }
            else if (F[i][j][k] == neighbor[1]){
                z = "-" + "-" + c.charAt(k-1) + z;
                k--;
            }
            else if (F[i][j][k] == neighbor[2]){
                z = "-" + b.charAt(j-1) + "-" + z;
                j--;
            }
        }
        while (i > 0 && k > 0){
            int neighbor[] = new int[3];
            neighbor[0] = F[i-1][j][k-1] + get_score(a.charAt(i-1),
c.charAt(k-1)) + 2*score[2];
            neighbor[1] = F[i][j][k-1] + 2*score[2];
            neighbor[2] = F[i-1][j][k] + 2*score[2];
            if (F[i][j][k] == neighbor[0]){
                z = a.charAt(i-1) + "-" + c.charAt(k-1) + z;
                i--;
                k--;
            }
            else if (F[i][j][k] == neighbor[1]){
                z = "-" + "-" + c.charAt(k-1) + z;
                k--;
            }
            else if (F[i][j][k] == neighbor[2]){
                z = a.charAt(i-1) + "-" + "-" + z;
                i--;
            }
        }
        while (i > 0 && j > 0){
```

```
            int neighbor[] = new int[3];
            neighbor[0] = F[i-1][j-1][k] + get_score(a.charAt(i-1),
b.charAt(j-1)) + 2*score[2];
            neighbor[1] = F[i][j-1][k] + 2*score[2];
            neighbor[2] = F[i-1][j][k] + 2*score[2];
            if (F[i][j][k] == neighbor[0]){
                z = a.charAt(i-1) + "" + b.charAt(j-1) + "-" + z;
                i--;
                j--;
            }
            else if (F[i][j][k] == neighbor[1]){
                z = "-" + b.charAt(j-1) + "-" + z;
                j--;
            }
            else if (F[i][j][k] == neighbor[2]){
                z = a.charAt(i-1) + "-" + "-" + z;
                i--;
            }
        }
        while (k > 0){
            z = "-" + "-" + c.charAt(k-1) + z;
            k--;
        }
        while (j > 0){
            z = "-" + b.charAt(j-1) + "-" + z;
            j--;
        }
        while (i > 0){
            z = a.charAt(i-1) + "-" + "-" + z;
            i--;
        }
        String d = "";
        String e = "";
        String f = "";
        for (int index = 0; index < z.length()/3; index++){
            d = d + z.charAt(3*index);
            e = e + z.charAt(3*index+1);
            f = f + z.charAt(3*index+2);
        }
        String s[] = {d,e,f};
        return s;
    }

    public static String[] multiple_sequence_alignment(String a[]){
        int score[][] = get_Needleman_Wunsch_pairs_score(a);

        // mendapatkan jumlah skor setiap baris
        int row_score[] = new int[a.length];
        for (int i = 0; i < a.length; i++){
            for (int j = 0; j < a.length; j++){
                row_score[i] = row_score[i] + score[i][j];
            }
        }

        // mendapatkan posisi baris yang jumlah skornya maksimum
        int star = 0;
```

```
        int max = row_score[0];
        for (int i = 1; i < a.length; i++){
            if (row_score[i] > max){
                max = row_score[i];
                star = i;
            }
        }

        if (star != 0){
            String b = a[star];
            a[star] = a[0];
            a[0] = b;
        }

        String s[] = sequence_alignment(a[0], a[1]);
        for (int i = 2; i < a.length; i++){
            s = sequence_alignment(s, a[i]);
        }


        if (star != 0){
            String b = s[star];
            s[star] = s[0];
            s[0] = b;
            b = a[star];
            a[star] = a[0];
            a[0] = b;
        }

        return s;
    }

    public static int[][] get_Needleman_Wunsch_pairs_score(String a[]){
        int score[][] = new int[a.length][a.length];
        for (int i = 0; i < a.length-1; i++){
            for (int j = i+1; j < a.length; j++){
                score[i][j] = get_Needleman_Wunsch_score(a[i], a[j]);
                score[j][i] = score[i][j];
            }
        }
        return score;
    }

    public static int get_Needleman_Wunsch_score(String a, String b){
        int F[][] = new int[a.length()+1][b.length()+1];
        for (int i = 1; i < a.length()+1; i++){
            F[i][0] = F[i-1][0] + score[2];
        }
        for (int i = 1; i < b.length()+1; i++){
            F[0][i] = F[0][i-1] + score[2];
        }
        for (int i = 1; i < a.length()+1; i++){
            for (int j = 1; j < b.length()+1; j++){
                F[i][j] = Math.max(F[i-1][j-1] + get_score(a.charAt(i-1),
b.charAt(j-1)),
                                   Math.max(F[i-1][j] + score[2],
```

```
                                            F[i][j-1] + score[2]));
            }
        }
        return F[a.length()][b.length()];
    }

    public static String[] sequence_alignment(String a[], String b){
        int m = a.length;
        int n1 = a[0].length();
        int n2 = b.length();
        int F[][] = new int[n1+1][n2+1];
        for (int i = 1; i < n1+1; i++){
            String s = "";
            for (int k = 0; k < m; k++){
                s = s + a[k].charAt(i-1);
            }
            F[i][0] = F[i-1][0] + get_new_score(s, '-');
        }
        for (int i = 1; i < n2+1; i++){
            F[0][i] = F[0][i-1] + score[2];
        }
        for (int i = 1; i < n1+1; i++){
            for (int j = 1; j < n2+1; j++){
                int neighbor[] = new int[3];

                String s = "";
                for (int k = 0; k < m; k++){
                    s = s + a[k].charAt(i-1);
                }

                neighbor[0] = F[i-1][j-1] + get_new_score(s, b.charAt(j-
1));
                neighbor[1] = F[i-1][j] + get_new_score(s, '-');
                neighbor[2] = F[i][j-1] + score[2];
                F[i][j] = get_max(neighbor);
            }
        }

        return back_track(F, a, b);
    }

    public static String[] back_track(int F[][], String a[], String b){
        int i = a[0].length();
        int j = b.length();
        int n = a.length;

        String c[] = new String[n+1];
        for (int k = 0 ; k < n+1; k++){
            c[k] = "";
        }

        while (i > 0 || j > 0){
            if (j == 0){
                for (int k = 0; k < n; k++){
                    c[k] = a[k].charAt(i-1) + "" + c[k];
                }
```

```
                c[n] = "-" + c[n];
                i--;
            }
            else if (i == 0){
                for (int k = 0; k < n; k++){
                    c[k] = "-" + c[k];
                }
                c[n] = b.charAt(j-1) + "" + c[n];
                j--;
            }
            else{
                String s = "";
                for (int k = 0; k < a.length; k++){
                    s = s + a[k].charAt(i-1);
                }

                int neighbor[] = new int[3];
                neighbor[0] = F[i-1][j-1] + get_new_score(s, b.charAt(j-
1));
                neighbor[1] = F[i-1][j] + get_new_score(s, '-');
                neighbor[2] = F[i][j-1] + score[2];
                if (F[i][j] == neighbor[0]){
                    for (int k = 0; k < n; k++){
                        c[k] = a[k].charAt(i-1) + "" + c[k];
                    }
                    c[n] = b.charAt(j-1) + "" + c[n];
                    i--;
                    j--;
                }
                else if (F[i][j] == neighbor[1] && F[i][j] ==
neighbor[2]){
                    double r = Math.random();
                    if (r < 0.5){
                        for (int k = 0; k < n; k++){
                            c[k] = a[k].charAt(i-1) + "" + c[k];
                        }
                        c[n] = "-" + c[n];
                        i--;
                    }
                    else {
                        for (int k = 0; k < n; k++){
                            c[k] = "-" + c[k];
                        }
                        c[n] = b.charAt(j-1) + "" + c[n];
                        j--;
                    }
                }
                else if (F[i][j] == neighbor[1]){
                    for (int k = 0; k < n; k++){
                        c[k] = a[k].charAt(i-1) + "" + c[k];
                    }
                    c[n] = "-" + c[n];
                    i--;
                }
                else if (F[i][j] == neighbor[2]){
                    for (int k = 0; k < n; k++){
```

```
                    c[k] = "-" + c[k];
                }
                c[n] = b.charAt(j-1) + "" + c[n];
                j--;
            }
        }
    }
    return c;
}

public static int get_max(int a[]){
    int max = a[0];
    for (int i = 1; i < a.length; i++){
        if (max < a[i]){
            max = a[i];
        }
    }
    return max;
}

public static int get_new_score(String a, char b){
    int s = 0;
    for (int i = 0; i < a.length(); i++){
        if (a.charAt(i) == '-' && b == '-'){
            s = s + gap_with_gap_score;
        }
        else if (a.charAt(i) == '-' || b == '-'){
            s = s + score[2];
        }
        else if (a.charAt(i) != b){
            s = s + score[1];
        }
        else {
            s = s + score[0];
        }
    }
    return s;
}

public static String[] sequence_alignment(String a, String b){
    int F[][] = new int[a.length()+1][b.length()+1];

    for (int i = 1; i < a.length()+1; i++){
        F[i][0] = F[i-1][0] + score[2];
    }
    for (int i = 1; i < b.length()+1; i++){
        F[0][i] = F[0][i-1] + score[2];
    }
    for (int i = 1; i < a.length()+1; i++){
        for (int j = 1; j < b.length()+1; j++){
            F[i][j] = Math.max(F[i-1][j-1] + get_score(a.charAt(i-1),
b.charAt(j-1)),
                               Math.max(F[i-1][j] + score[2],
                                        F[i][j-1] + score[2]));
        }
    }
```

```
        return back_track(F, a, b);
    }
    public static int get_score(char a, char b){
        if (a == b)
            return score[0];
        else
            return score[1];
    }
}
```

Lampiran 12 *Source Code* dari Data.java dalam *Package* MultipleSequenceAlignment

```
package MultipleSequenceAlignment;

public class Data {
    private static int population_size;
    private static int max_iteration;
    private static int number_of_sequence;
    private static int length_of_sequence[];
    private static int max_length_of_sequence;
    private static String sequence[];

    public static void set_sequence(String a[]){
        sequence = a;
        number_of_sequence = sequence.length;
        length_of_sequence = new int[number_of_sequence];
        max_length_of_sequence = 0;
        for (int i = 0 ; i < number_of_sequence ; i++){
            length_of_sequence[i] = sequence[i].length();
            if (max_length_of_sequence < length_of_sequence[i])
                max_length_of_sequence = length_of_sequence[i];
        }
    }

    public static String[] get_sequence(){
        return sequence;
    }

    public static char get_sequence_char_at(int i, int j){
        return sequence[i].charAt(j);
    }

    public static int get_number_of_sequence(){
        return number_of_sequence;
    }

    public static int get_length_of_sequence(int a){
        return length_of_sequence[a];
    }

    public static int get_max_length_of_sequence(){
        return max_length_of_sequence;
    }

    public static void set_max_iteration(int value){
        max_iteration = value;
    }

    public static int get_max_iteration(){
        return max_iteration;
    }

    public static void set_population_size(int value){
```

```
        population_size = value;
    }

    public static int get_population_size(){
        return population_size;
    }
}
```

Lampiran 13 *Source Code* dari Solution.java dalam *Package* MultipleSequenceAlignment

```
package MultipleSequenceAlignment;

public class Solution {
    String s[];
    int f;

    Solution(){
        s = new String[Data.get_sequence().length];

        for (int i = 0; i < s.length; i++){
            s[i] = Data.get_sequence(i);
        }
        int a;
        if (Math.random() < 0.5){
            a = Data.get_max_length_of_sequence();
        }
        else {
            a = 1 +
Data.get_max_length_of_sequence()+(int)(0.2*Math.random()*Data.get_l());
        }

        for (int i = 0; i < s.length; i++){
            for (int j = 0; j < a-Data.get_length_of_sequence(i); j++){
                int b = (int)(Math.random()*(s[i].length()+1));
                s[i] = s[i].substring(0, b) + "-" + s[i].substring(b);
            }
        }

        if (Math.random() < 0.1){
            s =
NeedlemanWunsch.multiple_sequence_alignment(Data.get_sequence());
        }

        f = Score.count(s);
    }

    Solution(Solution a){
        s = a.get_s();
        f = a.get_f();
    }

    Solution(Solution a, int type, int type2){
        s = new String[Data.get_sequence().length];
        for (int i = 0; i < s.length; i++){
            s[i] = a.get_s(i);
        }
        int b = (int)(Math.random()*s.length);
        int c = s[b].length()-Data.get_sequence(b).length();
        if (c > 0){
            int d = 1 + (int)(Math.random()*c);
```

```
        int e = 0;
        int i = 0;
        while (i < s[b].length() && e != d){
            if (s[b].charAt(i) == '-'){
                e++;
            }
            i++;
        }
        s[b] = s[b].substring(0, i-1) + s[b].substring(i);
        int f = (int)(Math.random()*(s[b].length()+1));
        s[b] = s[b].substring(0, f) + "-" + s[b].substring(f);
    }
    f = Score.count(s);
}

Solution(Solution a, int type, int type2, int type3){
    s = new String[Data.get_sequence().length];
    for (int i = 0; i < s.length; i++){
        s[i] = a.get_s(i);
        int b = (int)(Math.random()*(s[i].length()+1));
        s[i] = s[i].substring(0, b) + "-" + s[i].substring(b);
    }
    f = Score.count(s);
}

Solution(Solution a, Solution b){
    s = new String[Data.get_sequence().length];

    int c = 1 + (int)(Math.random()*a.get_s(0).length()-1);
    for (int i = 0; i < s.length; i++){
        s[i] = a.get_s(i).substring(0, c);
    }
    int d[] = new int[s.length];
    for (int i = 0; i < s.length; i++){
        d[i] = 0;
        for (int j = 0; j < c; j++){
            if (s[i].charAt(j) != '-'){
                d[i]++;
            }
        }
    }
    for (int i = 0; i < s.length; i++){
        int e = 0;
        int j = 0;
        while (j < b.get_s(i).length() && e != d[i]+1){
            if (b.get_s(i).charAt(j) != '-'){
                e++;
            }
            j++;
        }
        if (d[i] == Data.get_sequence(i).length()){

        }
        else{
            s[i] = s[i] + b.get_s(i).substring(j-1);
        }
```

```
    }
    int e = s[0].length();
    for (int i = 1; i < s.length; i++){
        if (e < s[i].length()){
            e = s[i].length();
        }
    }
    for (int i = 0; i < s.length; i++){
        String f = "";
        for (int j = 0; j < e - s[i].length(); j++){
            f = f + "-";
        }
        s[i] = s[i].substring(0, c) + f + s[i].substring(c);
    }
    f = Score.count(s);
}

Solution(Solution a, Solution b, int type){
    s = new String[Data.get_sequence().length];
    for (int i = 0; i < s.length; i++){
        s[i] = "";
    }
    int c[][] = new int[s.length][a.get_s(0).length()];
    int d[][] = new int[s.length][b.get_s(0).length()];
    for (int i = 0; i < s.length; i++){
        if (a.get_s(i).charAt(0) == '-'){
            c[i][0] = 0;
        }
        else {
            c[i][0] = 1;
        }
        for (int j = 1; j < a.get_s(0).length(); j++){
            if (a.get_s(i).charAt(j) == '-'){
                c[i][j] = c[i][j-1];
            }
            else {
                c[i][j] = c[i][j-1] + 1;
            }
        }
        if (b.get_s(i).charAt(0) == '-'){
            d[i][0] = 0;
        }
        else {
            d[i][0] = 1;
        }
        for (int j = 1; j < b.get_s(0).length(); j++){
            if (b.get_s(i).charAt(j) == '-'){
                d[i][j] = d[i][j-1];
            }
            else {
                d[i][j] = d[i][j-1] + 1;
            }
        }
        for (int j = a.get_s(0).length()-1; j > 0; j--){
            if (c[i][j] == c[i][j-1]){
                c[i][j] = 0;
```

```
        }
    }
    for (int j = b.get_s(0).length()-1; j > 0; j--){
        if (d[i][j] == d[i][j-1]){
            d[i][j] = 0;
        }
    }
    int e = -1;
    for (int j = 0; j < a.get_s(0).length(); j++){
        if (c[i][j] == 0){
            c[i][j] = e;
            e--;
        }
    }
    e = -1;
    for (int j = 0; j < b.get_s(0).length(); j++){
        if (d[i][j] == 0){
            d[i][j] = e;
            e--;
        }
    }
}
String F[] = new String[a.get_s(0).length()];
String g[] = new String[b.get_s(0).length()];
for (int i = 0; i < F.length; i++){
    F[i] = "";
    for (int j = 0; j < s.length; j++){
        F[i] = F[i] + "" + c[j][i];
    }
}

for (int i = 0; i < g.length; i++){
    g[i] = "";
    for (int j = 0; j < s.length; j++){
        g[i] = g[i] + "" + d[j][i];
    }
}

String v[] = new String[s.length];
String w[] = new String[s.length];
for (int k = 0; k < s.length; k++){
    v[k] = a.get_s(k);
    w[k] = b.get_s(k);
}

for (int i = F.length-1; i >= 0; i--){
    for (int j = g.length-1; j >= 0; j--){
        if (F[i].equals(g[j])){
            String x[] = new String[s.length];
            String y[] = new String[s.length];
            for (int k = 0; k < s.length; k++){
                x[k] = v[k].substring(i);
                y[k] = w[k].substring(j);
                v[k] = v[k].substring(0,i);
                w[k] = w[k].substring(0,j);
            }
```

```
                if (Score.count(x) > Score.count(y)){
                    for (int k = 0; k < s.length; k++){
                        s[k] = x[k] + s[k];
                    }
                }
                else {
                    for (int k = 0; k < s.length; k++){
                        s[k] = y[k] + s[k];
                    }
                }
            }
        }
    }
    if (Score.count(v) > Score.count(w)){
        for (int k = 0; k < s.length; k++){
            s[k] = v[k] + s[k];
        }
    }
    else {
        for (int k = 0; k < s.length; k++){
            s[k] = w[k] + s[k];
        }
    }
    f = Score.count(s);
}
public String[] get_s(){
    return s;
}

public String get_s(int i){
    return s[i];
}

public int get_f(){
    return f;
}

public int get_fitness(){
    return f;
}

public void print_solution(){
    for (int i = 0; i < s.length; i++){
        System.out.println(s[i]);
    }
    System.out.println();
}

public void print2_solution(){
    for (int i = 0; i < s.length; i++){
        for (int j = 0; j < s[0].length(); j++){
            System.out.print(s[i].charAt(j) + "\t");
        }
        System.out.println();
    }
    System.out.println();
```

```
    }

    public void print_score(){
        System.out.println(f);
    }
}
```

Lampiran 14 *Source Code* dari Population.java dalam *Package* MultipleSequenceAlignment

```
package MultipleSequenceAlignment;

public class Population {
    private Solution solution[];
    private int population_size;

    public Population(int n, boolean b){
        solution = new Solution[n];
        population_size = n;
        if (b == true){
            for (int i = 0; i < n; i++){
                solution[i] = new Solution();
            }
            sort();
        }
    }

    public Population(Population p){
        population_size = p.get_population_size();
        solution = new Solution[population_size];
        for (int i = 0; i < population_size; i++){
            solution[i] = new Solution(p.get_solution(i));
        }
        sort();
    }

    public void save_solution(int i, Solution s){
        solution[i] = s;
    }

    public void sort(){
        for(int i = 1; i < population_size; i++){
            for(int j = i; j > 0 && solution[j].get_fitness() >
solution[j-1].get_fitness(); j--){
                Solution dummy = solution[j-1];
                solution[j-1] = solution[j];
                solution[j] = dummy;
            }
        }
    }

    public Solution get_solution(int index) {
        return solution[index];
    }

    public Solution getFittest() {
        return solution[0];
    }

    public int get_population_size() {
```

```
        return population_size;
    }

    public void print_all_score(){
        for (int i = 0; i < population_size; i++){
            solution[i].print_score();
        }
        System.out.println();
    }

    public void print_all_solution(){
        for (int i = 0; i < population_size; i++){
            solution[i].print_solution();
        }
        System.out.println();
    }

    public void print_all_length(){
        String a[] = solution[0].get_s();
        for (int i = 0; i < population_size; i++){
            System.out.println(a[0].length());;
        }
        System.out.println();
    }
}
```

Lampiran 15 *Source Code* dari FireflyAlgorithm.java dalam *Package* MultipleSequenceAlignment

```
package MultipleSequenceAlignment;

public class FireflyAlgorithm {

    public static Population iteration(Population population){
        Population new_population = new Population(population);
        Solution new_solution[] = new
Solution[population.get_population_size()];

        for (int i = 0; i < population.get_population_size(); i++){
            for (int j = 0; j < population.get_population_size(); j++){
                int type = 1;
                new_solution[i] = new
Solution(population.get_solution(i), type);
                if (new_solution[i].get_fitness() >
new_population.get_solution(i).get_fitness()){
                    new_population.save_solution(i, new_solution[i]);
                }
            }
        }

        new_population.sort();
        return new_population;
    }
}
```

Lampiran 16 *Source Code* dari CuckooSearch.java dalam *Package* MultipleSequenceAlignment

```
package MultipleSequenceAlignment;

public class CuckooSearch {
    private static double pa = 0.5;

    public static Population iteration(Population population){
        Population new_population1 = new
Population(Data.get_population_size(), false);
        Population new_population2 = new
Population(2*Data.get_population_size(), false);

        for (int i = 0; i < population.get_population_size(); i++){
            int type = 2;
            new_population2.save_solution(i, population.get_solution(i));

new_population2.save_solution(population.get_population_size()+i, new
Solution(population.get_solution(i), type));
        }

        new_population2.sort();

        for (int i = 0; i < population.get_population_size(); i++){
            new_population1.save_solution(i,
new_population2.get_solution(i));
        }

        for (int i = 0; i < population.get_population_size(); i++){
            if (Math.random() < pa){
                int type = 3;
                Solution new_solution = new
Solution(new_population1.get_solution(i), type);
                new_population1.save_solution(i, new_solution);
            }
        }

        new_population1.sort();

        return new_population1;
    }
}
```

Lampiran 17 *Source Code* dari FlowerPollinationAlgorithm.java dalam *Package* MultipleSequenceAlignment

```
package MultipleSequenceAlignment;

public class FlowerPollinationAlgorithm {
    private static double p = 0.1;

    public static Population iteration(Population population){
        Population new_population1 = new
Population(population.get_population_size(), false);
        Population new_population2 = new
Population(2*Data.get_population_size(), false);

        for (int i = 0; i < population.get_population_size(); i++){
            if (Math.random() < p){
                int type = 2;
                new_population1.save_solution(i, new
Solution(population.get_solution(i), type));
            }
            else{
                int type = 3;
                new_population1.save_solution(i, new
Solution(population.get_solution(i), type));
            }
        }

        for (int i = 0; i < population.get_population_size(); i++){
            new_population2.save_solution(i, population.get_solution(i));

new_population2.save_solution(population.get_population_size()+i,
new_population1.get_solution(i));
        }

        new_population2.sort();

       for (int i = 0; i < population.get_population_size(); i++){
            new_population1.save_solution(i,
new_population2.get_solution(i));
        }
        return new_population1;
    }
}
```

Lampiran 18 *Source Code* dari GenerateSequence.java dalam *Package* MultipleSequenceAlignment

```
package MultipleSequenceAlignment;

public class GenerateSequence {

    public static void main(String[] args) {

        int n = 15;
        int l = 5000;
        String sequence[] = SequenceGenerator.generate(n, l);

        System.out.println("private static String s" + n + "" + l +"[] =
");

        System.out.println("        {\"" + sequence[0] + "\"");
        for (int i = 1; i < n-1; i++){
            System.out.println("        ,\"" + sequence[i] + "\"");
        }
        System.out.println("        ,\"" + sequence[n-1] + "\"};");
    }
}
```

Lampiran 19 *Source Code* dari Compare3Sequence.java dalam *Package* MultipleSequenceAlignment

```
package MultipleSequenceAlignment;

public class Compare3Sequence {

    public static void main(String[] args) {
        int n = 10;
        int l = 250;
        String sequence[] = Sequence.get_sequence(n, l);
        /*
        long startTime = System.currentTimeMillis();
        String a[] = NeedlemanWunsch.alignment(sequence[0], sequence[1],
sequence[2]);
        long endTime = System.currentTimeMillis() - startTime;
        System.out.println("Needleman-Wunsch Alignment");
        System.out.print("Hasil alignment   : " + a[0] + "\n");
        for (int i = 1; i < 3; i++){
            System.out.println("\t\t    " + a[i]);
        }
        System.out.println("Skor alignment    : " + Score.count(a));
        System.out.println("Panjang alignment : " + a[0].length());
        System.out.println("Waktu     komputasi              :     "     +
(double)(endTime)/1000);
        */
        long startTime = System.currentTimeMillis();
        String a[] = Star.alignment(sequence);
        long endTime = System.currentTimeMillis() - startTime;
        System.out.println("\n\nStar Alignment");
        System.out.print("Hasil alignment   : " + a[0] + "\n");
        for (int i = 1; i < n; i++){
            System.out.println("\t\t    " + a[i]);
        }
        System.out.println("Skor alignment    : " + Score.count(a));
        System.out.println("Panjang alignment : " + a[0].length());
        System.out.println("Waktu     komputasi              :     "     +
(double)(endTime)/1000);
        startTime = System.currentTimeMillis();
        a = NeedlemanWunsch.multiple_sequence_alignment(sequence);
        endTime = System.currentTimeMillis() - startTime;
        System.out.println("\n\nMetode Baru Needleman-Wunsch Alignment");
        System.out.print("Hasil alignment   : " + a[0] + "\n");
        for (int i = 1; i < n; i++){
            System.out.println("\t\t    " + a[i]);
        }
        System.out.println("Skor alignment    : " + Score.count(a));
        System.out.println("Panjang alignment : " + a[0].length());
        System.out.println("Waktu     komputasi              :     "     +
(double)(endTime)/1000);
    }
}
```

Lampiran 20 *Source Code* dari NewMain.java dalam *Package* MultipleSequenceAlignment

```
package MultipleSequenceAlignment;

public class NewMain {

    public static void main(String[] args) {
        int n = 3;
        int l = 5000;
        String sequence[] = Sequence.get_sequence(n, l);

        int population_size = 10;
        int max_iteration = 1000;

        Data.set_sequence(sequence);
        Data.set_population_size(population_size);
        Data.set_max_iteration(max_iteration);

        long startTime = System.currentTimeMillis();
        Population population = new Population(population_size, true);
        for (int i = 0; i < max_iteration; i++){
            population = FireflyAlgorithm.iteration(population);
            //population = CuckooSearch.iteration(population);
            //population                                                =
FlowerPollinationAlgorithm.iteration(population);
        }
        long endTime = System.currentTimeMillis() - startTime;
        String a[] = population.getFittest().get_s();
        System.out.print("Hasil alignment   : " + a[0] + "\n");
        for (int i = 1; i < n; i++){
            System.out.println("\t\t    " + a[i]);
        }
        System.out.println("Skor    alignment                :    "    +
population.getFittest().get_f());
        System.out.println("Panjang alignment : " + a[0].length());
        System.out.println("Waktu     komputasi              :    "    +
(double)(endTime)/1000);
    }
}
```

Lampiran 21 *Source Code* dari Sequence.java dalam *Package* MultipleSequenceAlignment

```
package MultipleSequenceAlignment;

public class Sequence {

    private static String s1010[] =
        {"TCTAACCGG"
        ,"TCTAACAACGG"
        ,"TCTAAACGC"
        ,"CCGAACTCACGC"
        ,"TCCAGCCAGC"};

    private static String s1020[] =
        {"AGCCGGGATCCCACACGC"
        ,"ACTCGTGAGAGCTCAACTCACGCC"
        ,"ACCTGGGGGCTTCGACGCC"
        ,"GCCGGGGCATCCATCACTCGAC"
        ,"CAGCCGTGATTATTCTCG"};

    private static String s1030[] =
        {"CTTTAAGCCGGGCGATGAGCGAGACGCCCCTA"
        ,"CTTTTTACGGGAGGTACGGCAGCTTCCCCA"
        ,"CTAAGGAGCGGCAGGTCAGGCAACGCCGCCCT"
        ,"TTCTTTACGGGGCAGTAACCGGCCCCA"
        ,"CTTTTTGACGTCAGGTAGGCAGCGCCGCCA"};

    private static String s1040[] =
        {"CCATAAGAGGCGAGTTAAGTTTTAAACGTTAGCACTTTGT"
        ,"CCCTACAAGTTCGGTATAAGTTTAAACGACGTAGCGATCTTAT"
        ,"CACTGAGCGTATCTAAGGGTCTTAAAACGATGACGACTATT"
        ,"TATCAGACGGATGAGTTTTAAACGTTGCCGGGCTTTAT"
        ,"CCCTGGGAAGACGGATCATATTCTTTCAACCTGTTTCGCATCCTTAT"};

    private static String s1050[] =
        {"TCTGGTGAGTGAATTAGGGGTGTGTGCGTAACATAATTTTGTATCGT"
        ,"TTCGGTACGTGAATTCGGGTGGTCGCGTCGATACTATTCTTATGCA"
        ,"TACTGAGAGATGAGAATGGGGTTGGTCGACGTAACTTATTCTATTGA"
        ,"TTCGTGCTAGGTAAATTCCAGTGGTCTGTTGCCGGGACTCCTAATTTCATTGA"
        ,"CTCGAGTAGTGAAATTTCAGGTGGTGTGATCGCCGATCACTATTTCTCATTGGA"};

    private static String s1060[] =
        {"ATTCTTCAGAGGCCCAGAGCGAGATTATCATGAAGTGTAGTTCGGCAAAGGAATCCCACTAG"
        ,"TATCACAGGCCAACTGCAAGTTAATCATAGAGGTTAGCTGGGAACAGAGGTCTCGCATG"
        ,"TTTAATGGCCCAATGAAAGTTTTACATGGAGAGGATAAACTGGGAAGGACCCTCATG"
        ,"ATTCTCAGCCATACGGACTTACTTCAAGGATATTTGGAGAAAGGTTTCCCTCACGT"
        ,"ATTCTCATAGCCATAGCAGAATTTAGAGGTGAGGAGTAGCTGGGAAACGACTCTCATG"};

    private static String s1070[] =

{"TGGGTACCAAAATAGGATTTGGTGTGGCTCTCCGAGCTGATGGCGTCATTTAGTGTAAAGACTACACC"
```

```
,"TTGTTAGCGAAATGGGAATTGAATGGGCTTTTCCGGACCGGATGGCGTTTTCATTCAGGCTTGAAGACTAC
GTCC"

,"TGTTACGAACAGGAGAATTGGTTAGGCGTCCCCACGCGTGGCATTCATCGACTCAGCATACCAC"

,"GTTACTAGTCATAGGAATTGGTATGGGCTCTCGCCGACCGCGATGGCCGTTCTTCAGGAGTTAGCTACAAC
TC"

,"TGTTCAAGAACCAAGTTGGTAGAGCTCACCGGACCAGCGATGGCTTTCACTTCAGATGGTAAGACTAACGA
CT"};

    private static String s1080[] =

{"TTCAGAAGTATGCATCCGCAGTTTCCATCTGGCTGCTTGGACGCTCGCGGTATAAATAATGTCCCCACAGT
GATGAGTCTGCCTTGAT"

,"TTACGAAGTAGGGCTCCATGTCCATCTGCACGTTTCGACGTAGCTCATAAATAGACCCCCCCAAGGGAGTG
TCTGCCCGGTAAC"

,"TTACAGGAAGGGACTCGCGTTCACATCGGCTCGGTTTGGGCGTCGCCTGACAAATGATTACCCTCAAGATG
ATGGTACGTCGGCGTAC"

,"TTATGTGCTGACACTTCGGTCAATGCGGCCTGATTGGCAACGTCGAGTATAAAATGATTCCCCCAATGGAT
GGTCTGCCGGTTAAC"

,"TTACCAGGTGACCCTCGCGTCGATCGGCCTCGTCTTGCACGCTTTGTAAGTGTTATCCCCACAAGGTGGTC
TTCCTGGTAA"};

    private static String s1090[] =

{"GCTTTGGATGTTCCCTGAAGAAGCCCAGGGTTAATGAACCTCGCGAGGGATTTTAGGCAAGTTGGGGCGTC
GTCGCAAAACATGCGTATACTCG"

,"GCTTTGAAGTTCCCGAGAAGAAGGGTATAGAACGACTTCGGGGTGGCTTTTAGAAGGGCGTACGCTCAAAC
AAAGCGATATTCG"

,"TGATTTTCCCGAGACAGCAGGGATTAATACCGATTGCGGTAGGTATTTAGGTGGGCTTGCTCTGGAAACAA
CGTAATTGCTG"

,"GGTTGAATTGTCTCCGAGAGCCAACGGGAATAATGAACGCAACCTGGGGTGTTGATCTCAGGAGATGGCGA
GTGCTCCGAGTACAAGCCGAGTTTTAG"

,"GTTTCGAAGTTTCGGAAGACGCCGTGGATGATGAACCACTCGGGTGGTGATTTAGGACATGGGCAGTAGTT
CCGAGCAACCAGCAATTATCGA"};

    private static String s10100[] =

{"GCATTTCGCCGTTACTAATATCACCGGACTCAGGGGACTTGGTCTGCCCTATGACGCGGCCCTACTCGCCA
TACTAAAGAATTAGAATACGGAAA"

,"GAGCTCCTGCTTAATTCATACTACCCGAACCTTCCGCTGACTTGAGTTCGCCCAGTTTCGACGGCTGCCCA
TCCGCCACTAAAAAGAATAGCGATACTGTCAA"

,"GAATTCCGTCGTTTATTACCGATACACGCGACTCTGACGGCATTCGTGTCGCCCTTGTGCGCGCACATTCT
GCCATTATCGAAAAGATTGATCATTACTGCCAAC"
```

```
,"AGAGCTCTCGCCTATAATAACTCAGCGAGATTCAGCGCCTGAGTCGCCCCATTTGACGGCGCCCACCGGCC
GTCCAAAATGATATAGATACCGGAAAA"

,"CCGACTCGCTGTAGAACGTCTCGGACTATACCGACCATAGGTTCGGCCACTTGACGCGGACCGTTCGCCCA
TACACTAAAGTGTAGTACACGAAA"};

    private static String s10110[] =

{"TGGTACATCCTGTGACCTGACGTGTGCGCCTGTTGTCTCCGCAACGCCTGCACACAACAACAAACATTCGA
GCTGCGGGATATTACCCGGCACGTCTCGGAGAGACCG"

,"TTGTCAAGCTTGGACCTAAGTCTTGCCGGTCGTGACCGGAGACCGTCTTGCAAAAAAAAGACATGTCGATC
GCCGACAATCTAACTCGACCTTCGCGCGATGAACG"

,"TGTCCGACCTTTAGCCTGAGTTTGTGGTCGGTTCACAGACCGCTTCGCACAAACAACAACATCGGACGCTC
CGAGAACTTACCCATGCACTGCCGCCGGCTAAAGCAG"

,"TAGTTAACGCTTTAATCTCGAGTTGTCGTGCTGGTACCGAGGCCGTCTCCAGAAAAAGAAAACTTAGATCC
GCGAGGACCTTAGTCATCGCTGGCGCGTGAGCAACCG"

,"GCAGACCCGTGACCTGGTTGCCATCTTGTACCGAACCCGTTCCGCAAGAAAAACATACTATAGATTCCGCG
AAGTCAGTACACCCCACGGCTTGCGGCGAATAAACAG"};

    private static String s10120[] =

{"CGTTAAACTGATTGCCATCCCAACGAGGCTTAGGTCCGCTGCATAATAAGTTCTCTTAGCTGACCTCTGAT
ATAAAAACGTACGTGCGTACGCTAGTAGCACTATCAAGACTGTGAG"

,"CGTATGTAACAGATCGACTCCGCCCAATAGCGGACTGCCTCGACAAACTCAGGGTTCTTAGCGCACCCTCG
ATATGATAGACCGACTGCGTGAGCGTTAGTTCGAAATCAAACTTGGAA"

,"GGTTGACAACGGTAGTCGCTACTCCAAATGCGACCTGTGCCAGTGTTAACTACGGTATCTATAGCGACGCC
TCCTTGTGTATACAGTACAGGCTGACGTTAGGTCGAAAAATAACAGCTGTGA"

,"ACTTGAAACGTATGCCTAGCACCAATTGGCGTTTGCTCGCGCCTAAATCCTGAGTCTCTTGCGCCCCCTGT
TTAATGACAGATCTGGTAACGTTAGCTGAAAATGCAAGGCGCTTGTGA"

,"CGTGTAATAGTGATCTACTCCCATAGACGCTGTCCGCTGCCTATGCACGGTCTTCAACGACCCCTCTGGTC
TAAATAAAGTACGTTTCGACGTGTAGTATGAAAAATCAAGTCGGAG"};

    private static String s10130[] =

{"GCAATCTCCCTCGTTTTAAGCGTGCTGATCCGCCATACCCCCTATCCTTGTCGAGAAATCTAGTACTAATT
GTCTCGCAAAACCACATTGTCCAGGCTTGCGGCTCCATCTACGTCGACATATTA"

,"AACAATCTGCCACATTTATGCTCGCGCATTCGCCCATACTACCCTTCCCTTCCGAACAACCCTGTCTATAA
TTTGCCTCGTAAACCTCAATGTGCGAGAGGCATCGGCAGTCCACTGATGCCGAACACAT"

,"GAGCAATTCCCCACGTTTATGTGGGTCCATTGCCCTAATCCCCGGCATTCCGTCCGGAAAAACTGTTAGTA
ATTTGCTCCGATACTCAATGGTAGCGAAGCCAACTGGCAGTCCTCGTAGCCAACATTT"

,"GCCAACTTCCCGAGTTAGGGCATATTAGCCATCGCCGCTATCCTGGCGGCAAAAGGCTTCGTTTAATTATG
TGTCCGAATCAATTCAATTGGCGGAAGCGCCTCGCAACCCCGTCAGCGACATTA"
```

```
,"CTCAATATCCACTTAGGCTCGCTATTCCCATAGTCCTCAACTCCTGGCCGAATAAACGTCCTTAATTATTG
CTCAATAACGACTAATTGGCGAGGAGCGCGATCCACCTACGCAGACATA"};

    private static String s10140[] =

{"ATTTTACGAGAGTGCAATATTAACAATTCTCGATTTTAAGCCCGAGCGGTAGCCCGTTCAGATCTCTTTCT
AGGTCTATTCCAGAGCTAAAAGTAGGGTGCTATGACTAGCTTAACGGGACTTCTCTAATCTAGCTTGTTG"

,"ATATGTACCGAGGGCATTTACAGAATTGGTGCTATTTTACGCGGGATGCTGCCGTGGTCATGACTCTCACA
AGCAATTACAACCAAAAAAAGTGAGGGCGCTTTACATTCTTACAGGCGACTTAGTAACTCGCGTGGTGA"

,"CTTTTCCACCATGGCATATGAGCAAATTTGGCCGATCTTTTACTGGGACGTAGCCGTGTACAGTACTTCCT
ACAAGCAAGCTGCTACTAAAAAAGTAGGGGCTATTTAGCGACTATCAGTGCACTTCTAGTAATTGCTTGTT"

,"ATTGGCCGCTAGGCAAATTGAACGAATTGTCCCGCTGTACGCGGCAGCATGGTTCACATTTCACTAGAAAT
TCAGAACAAAAAATGAGAGGGCTAAATCAACGCTTGAACGCGGACGTTCTGATACTATGTCGTTTG"

,"ATTTTAGCAGCGCGATTTCAATATTGCCGTTTATAACGGTACTGACGTGGTCTAGCTTCTTCAGAATTAAA
TCATGAACCCTAAAAATGGAGGGCGTATTACGTTAACGGCGCTCTAGTAAGCTTGCTTGATG"};

    private static String s10150[] =

{"CGGACGCAGAGCTCGAGTAACAATTTGGGTCAGTATCATCTCATGCGTTTCAAAGTGATCGTCTCCGATCT
CGTGGGGGAAGTATCGAAAGCGAGTGAGAAATGGCCACACGTATGTGGAGAACTCTAAAGGTGTACTTTGTG"

,"CGGAGCGGGGACCTCGGTAACCAATTGGGGCAATGTCTACCAAGGCCTCCAAATGGTAAGGCTACCCTACT
AGCACTGGAGGAGAGTATATCCGATTGCTAAGGTTGATGGCTACTCACGATGTGGAAGACGATAAGAGATACT
TTTGCG"

,"CGGAAGCAGACTCTTACGGTAAAAATGGGCGAGTATCTACACATGTCCTTTTCAAAGTGGAAGGCATACCC
GATCTGCCGCGGCAGAAGTGAATCGAAGTTCGAAGGGTAGATGGCCGCCCGCATTGGAGACAGATACATGACG
TACATTAGTC"

,"CGCTAGCGGCATTAGTAGACAATTGGGCAGTACTAACCTATTGCCATTCTGGTGTGACGGGGCTACCGGGA
GCTGCCGGTTGGAGAGCATTACGAGATTGCCAAGGTTGCTGGCACACCGAATAGGTAGACAGGTCAGGAGAGT
GCTTAAGTCG"

,"GCACAGGGACTTATGGTAGAAATGGGCGCATCTACTACAAAGTCCCACTAACAGTGTAGGCTACCCGATTC
GTCCCGTGGAGAAGTAAACTAGGACTTGCAAGGTCAATGACCAGCCTATGAAAGACGAATAGGGAGTATTAGT
GC"};

    private static String s10160[] =

{"GAGGCCCGATCATGTTTAAATCACGGTGGGCACGTTAGGAACACCAGAATAATAGAACCGTCGATTCCTCG
TGAAATACTGCTTCAGATTTCGCGTGTTAGTTGTCTCAGTGGGCTCGGTATGTAGTTCGTGTTAAACTTAATG
ACTATTTGGAAGTGGAGGG"

,"GGCCCGATCATGGAGATACATAGGGTGGCCGACATTGTAGGGGATCACAGAGATAAAGTGAAGCCGGGATT
TCCCGCTTAATAGTTCTCTGAATTGGGATGGTGGCTTCTTACTGGGGGGCTGCGTGTATGTGAGTTGCTTAAA
AGTTATGATTAAGTGAGATGAGAGTCG"

,"GACCTCAGTCTGTATAACTCAGGGGAGCACATTTAGGGACTACAGAGTAATGGAAACGCGCATTACCCCGC
TAGATTGCCTTAGATTTGGAGTTTTGTGCTTTATGAGGCTGGCGGTATTGAGTCTTAAGGTTATATATGTTGG
AGATCAGGGCG"
```

```
,"GCCCATCTAGCTTATAAGCCGGGGGGCCAATAGAGCCGACAGGACAAATGGAAACGCCTGATTACCCCGTG
AATGACGTTTAAAATTTCGGAGTTCTATGCTAATACTAGTGGGGCTCGGTAATTAGTGCTATAGTGAATATAG
TGTGAAGTGATCGCG"

,"GGTGCTCATTGTAGTAAAATCGGTGGCCAATTTAGGCGCCCAGAGACTAATAGAGAAACCGCGAATTGCCC
CCGGAATACGTTAGATTTGAAGTTTGGTATCTATGGAGGTGGCACGTGATTGTGGTTAAACGTCTATATTAGG
TGGACATGGGACGTG"};

    private static String s10170[] =

{"AAATAGTCTGCTGTGTTGACCGATAGTCCCGGGAGCCGCATGGGCGGCCGCAGTATCACCGTTAGGATTAG
AACAACAGTACAGAGCGACCACTCGTCAGTTACCTTGTCGTGCGAAGTAGCATCTCTCCCGATCATGTGGGCA
GGGCATGGTCCAATGCGTACCCGGGGATATT"

,"AATACCGTTGCGTACCGATAGTTCGGTAGGATCCCTCTTATGCGGCGCTGCGTTCAGACGCTCAGCTGTAG
ATAAAACAGTCCGGCACTCAACTGCCGACATTTGCTGTAGTAGTAATTTGAATGCCCGGCGGTGGCTGGCTGT
GTCAAACGATCGCTGGTGCTT"

,"AATTCCTTGGGCTTGCTACCAGTAGCCGTTTGACCGTACTTGGCGGTCAGCGGCAGTATAACGTCCCGGGT
AGAATAAGCTTAGAGTCTTACTACCTGTCGACGTTTCCTCTGGGAATTTATCAACTCCGCCACGGAGTGCTTG
GGGTCGGGCATGTGGTACTACAGACGCGAGGATT"

,"AACTACCTGGGGCTATGACCGATTGTTCACGTGATGCCATTGCAGCGGCGGGACTTCAACCGCCCCGCTAT
GAATAAGGGTTCCGACGATGCCTCGTGAACCTGACTTGCTCTGTGATATTACACACCGCGACCTGAGTCAGAT
GGGCTGGCATATCAATACGATCTGCGGGGGATT"

,"AATATCGGAGGGCCGTGCTCACGATATGCCGTAGGCCGCCGTTCGCGGCGCGGGCGTTACACCGCCGTGTA
GGTAAACGTCCGAAGCTCCTACCGTCTGACCTTGCTTGTCGGTGATTTTACACCTCGCCCGAATCAGTGGTAG
CTAGGTAACATAACGAGCGACGTGATT"};

    private static String s10180[] =

{"TTAAAAATCTTGTTATGTACTCTTGTCCCTGGGGACGGGGGGCAACAGGCGCTAATGTAGAGCTCTCGTAT
GCTTCGTAAATGATATATGACCAGGTCGTGCGTAAGACGACCTCCGACTCGCGCTACACATTTCTGGTGCGTT
TCTAGTGCCTGTCATCATCGACGTGGTGGACCAACCTATT"

,"CAAAAAACTGTAAGCCAAGTCCTGTACCCGGGGAGAGAGGGAGGGCCGGACGGATCTAGGGTCGTGTCAGT
TCGAATATGTATCATAGCCGCCTGGCTAGATCGCATCCAGATGCCGCGAACACCAACTCCTAGGTCTCGTTCA
CTCCCGTTTAGTCAGCTGGTGACCAGTCT"

,"TTAAAAAACTCGTTCTTTACAGGTCCTATATCGGGAGGGTGTGGAAAGACATTAGTCAAGGGGCTGCCGAC
TAGCATAATTAATCATGCATACCTGTATGCTTAGCGGTACTTCAGCTTACCACATCAATTCTGGATCTGCTTC
GAGTCGGACAATCTCACGTGGACCACCT"

,"TTAAGAAAATCATGTTATCCAGTCCCTTGATGCGCGGGAGGTGGGAGAAGGCCCAAATGAGCGTCCGGCTA
CCTTGTATTAGTTAGATCGATGCAAGCGCTGTTTTGAATCGCATCCTGACTAGCAAACGCAATTCATGGTTTC
GTCCTTGAACGCTGTACATCTACTCGCCTGGGGACCACAT"

,"ATCAAAAATCTTGATTGCCATAGATTCCTTTATCCGGGATGGTGGGGAAGACGCGATTGATCCAATGCGGC
GGTCAGCTAGCTTCAAAAGGTTAACTTGCAACGGTTGCTTGAAGCGACTCGACAGCCGCAAACCTAATTCCTG
GGTCTGCTCTGGTGGCGTCATTAACTCAAGTGAGCCACCCATT"};

    private static String s10190[] =

{"TCCGCATTTTTGCGGAATAGTTAGCCCCCAGATAATAGATGTATGGTTTGGATGGGCGATCTCGCGAGTGG
```

```
GGGGCGGAGTCTAACTAGGACCCCAATATTGCGCGGGTATACCCGACAGTAGTTAGCCCTAGCGATCCACGAT
AATCGTTTCCCCCTGGAAAGACGGCCATGTGAGCAATACGTATCG"

,"CTCGATTTTCGGCGGAATCAAAGGCACACGAAATAGTGGAAGTTTGAGTGAAGATGCATCTCGCCTTGGCC
ACGGGCTGTATCAGCTCAGGAACCCCCATTCCAGGATACTCCCAATGTAGGTTAGGCCTAGGACCACGGATAG
TATCTTACCCTCATGGAGAATGTAGCACGATGAGATTTTTTAATG"

,"CAGGTTCGTCGCGGACTATAGCCCCAGTACAATTATGGTAGGTTTTGGTAAGGTGCAATCCGCAGGTCGCA
GTGGCTGGCCACCTAGTAGACCCAACAATATGCCAAGCAAATGCTCGCCAAGTTGTTGAACTCTGGCTCCCAC
GGGATATAACTCTCCCCACGTACGAAGAGGCCATATGGGAAATCGAAG"

,"TCCAATTTGTCCGGGATAATTAGCCCCAGGCAAAGTTAGTAATAGTTGAGTAGTGCCGTCTCCGCAGTGCC
GAGGCGGTGATACCCTAGGAGGCCACCATTGCCAGCGGAAATCCCACGAATGATGACGTAGGTATTCTCACGA
GTAACACGTCCCCCATAGAGATATGACAATAGAGCAGCTATTTGATA"

,"TCCGTATTCGCGCGGATAACTTAAGCCCCCATGCAAATAGTTATTTGGACTAGGTTGGCTGCTGCAGGGTC
CAGGGTCAGTACTCTAAGGACCCACATTGCCAGCCGGGAATCCTCCCCCAAGATGTTGAGCTATGGATCGCAC
GGATGTACTACTTCCGCCATGGGAATAAGGACCGTATTGACATTGTGCATGT"};

    private static String s10200[] =

{"ATTTCTGGTGACTAGGGGCCAAGCGTGCCCGGACGACGTTCAGGCCTGTCCTTCGACGTTCTTCAATGTGC
GCCGATCTGGCTTAAGGGGTTATTCACGTATATTCGCGACACCGAAGTGGTTTTTTTACCCCGAAGTGCAATG
TTCTAAAACTAGATAATGCAGACGCGTTGGCTGTGACAATTAGGTATGGAGGTACCGCTGTTCGG"

,"TATTGATGGGTACAGGGGGCCATGCCCTCGAGGAGGGCAGTTCACCGTGGCCTACGGTTTCTCCAATTTCG
CCGGGTCAGGACATGAGGTGGAAGTTCAGTATACGTCCACGGAAGAGGTTGGAACCCTCGTATGAGAAGCTAA
ACATTGAAAAAGCCCAGAAGGTACGTGACTTGAATAATCTTGGTATCGATTGGG"

,"ATTTCGTGGGACGGGCGGCCATGCTTCGCCAGAGCTGTCATCTGGTCTGTAGGTATTCGAATATTGCGCAG
GCTTGGATAAGAGAGAGTTCATTATTACCGAGCACGAGACGGTTGTCACTCGCTAAGCGGAATCTTCTAAACA
TTATTAAGGCCAGACGTTCGTGACCATATAGGATTGTCGGTTGGCTGTTGG"

,"ATTTCGTGGTACGGGGGCCATGCCTTGGCGGGAGAGATTGCCGTCGGTCCTCGGTGCTTTCGTATTTGGCT
CGTGGTGTGCTAGATGCGATTCAGTAATATCGAAGAACTACGGGGCTGTTACCCTCGGTTAGAGATTTTACTA
AATAGGCAGAGGGCTGCGAAGGATATTGTTATGTGTGTGACTGTGG"

,"ATTTGCGTGGGTACGGGGGGCATGCTCTCCGGAGGACGCTTGCAGCTGGTCTCGAGGTCTTTCCGAAGTTG
CCGTGGTTGGCTAAGGGATATGAATCCACGAGCAGGAGTGGGTTGTGCCTCCTAACGGATTTTCTCAGACATA
ATTATGCAGACGGTTTGCGATAGCTAGTAGTAAGAGATGGCTGTGG"};

    private static String s10210[] =

{"GGTCACTTGTCAGGATCCGCGATTACCATGGCGGCGGACATGGTGTTCTCGAAGGTTGCAAAACGGGCCTT
CAATTCGGTACTTAGGAGGGGCTCAGGGAAACCTTTGAACAGATCTGCAGAATGAAAGGAGACAGCAGAAATT
TTCCAAGGGACCTCCTCCTGGGCGACCGCACGGACCCAGCACATCGAACTTCTCTTATCAGCAACCCCC"

,"GTCCCGAAAGGAGGCGGATTCGCCGGGGGCGCAATGTTGCAGGAAATGAAACTGTCTCTCATGTCGGTACC
TTCGAGGGGCCTGAAGGATCTCGTTGCTTCCTAATCAAGGAGTATTGATGCAGAGCTAGACATTTTGCCTGGC
AAATGAGACCTCCCTAGCGGGGACCGACAGCGAGACGCCATACAATTTCTAATCTGAGTATCATCGCC"

,"GTGCTCAGAAGAGCAGCGCAAGGCTGCGACGGCGAATGTGTCCTAGATAGAAGCCAGCCCTAAACTCTCGG
CATCTGAGGGAGCCATGGTAGTCCTTGAACATCGATTCACAGTAAGTTAATACAACGGGCAAACTATTCTCAT
AATGCACCCCTCCCTGCGGCGGAGCAAAGCCGACCGCATTACACCTCCTATAAGTCGTTGCATGATCAGGCC"

,"GTCGCGTAAGGACGCGCGTGATATCCCCGCCGCCAAGTGTGCCGAGACTTAAGACAGGCCCCTTTCGAATG
```

```
TTCGGTATCTGCGAGAGGCTCTTAGGGGCTCTTTGACTGGTATTCCTCAATAATTGAGAAACCGTGCAACCGT
TTCCACATGCCCCACGTCCGTCGAGCACGAAACATGCGAGCATTCTGCATCTTCAAGTATCGTAGGCAATACG
CCC"

,"GTGCCTGAAAGGACGCGTGATCTGCCGCTGCCGAGCGATATGTTCCGAGAACGTTGAAAACGCGCTTCAAA
TTCTGGGTAGCCTCGGAGGCGCTGAGTGCATTCCTGTAACTCTTGATTCTCAGTTGACTGGCAAGCGTGCCAA
CTTCTTAATAGCATCCTCTTCTCCGGCGCACTAAACGAGCACGCCATAAAACTGCCTATACATCGTAGCAATC
ACGCC"};

    private static String s10220[] =

{"TTGCGGACTCGATCCCACGTTCCCAGAGTTCGTTCGATACGGGAGGAGACGGATTCTGTCAACATTCAGGT
ACTCGGTCAAAGTTCAAGTCTAATAGTCATGAACGGAGTTCGTGTGACACCTTATACCACCCTGCCGTCTTGC
CTTAGCATAGTGAGATGTCCGGGCCCCGCGCCACGATTGAGTCCCGTACTCGCATAGCAGATTTGAAAAATGG
TGA"

,"TATGGGGAACTGACACTACCTCCTGTTCGTAGGTCGCGGGTAGATCCGATCCCTCGTAACTCACGCAAAAA
GTTCAGAATGTCGACACGAAATGCGCTAGCATGGAGAGCCGTCAGACAACTTAACTCCGCAATCCTCTCTTGA
CTTTAAGAGCTATGGTAGATGTGGGGGCCCCCTCCCATTTATGGCACGTACTCGCACGACAGTTTGAAAAACT
CGCGGA"

,"TTGGTCTGAATCCCATCCTTCGGTTTATGCATATGCGGAGAGGATAGCAGTCTCGTGCTAAAGCCTACGTA
ACAGTTTCGTATGTTCAATTCTAATGACCTAGGAAAGTGATAGCCTCTACACATCATACCCCCTCCCCTCGTC
TAAGCAAGTATGGGATGTGGGGGCCGCCCTACATTGGCTCTCTGCTCGAGGTCAGATTTGAAACGTTGCC"

,"TTGGGAGGATAGCTCCCACTCCTGTTCTCGGCTTAACGTGAGGGTCCAAGAGCCCCTTCGTCCCCTCACGT
AAGCTTCGAAGGTTAAGCCGAAGAGCCAAGAGGGTAACTCTTCAGCAACTTAACCCTCTTCTTCTTTGCTTGC
AATGTGATTGAGAGTGGGGGCCGCGCGCCCTCTGTTGCTCACGTCCTGACAGCGGGATGGTAAACCGTGGA"

,"TTGGAGGATTTGTCCCAACTCTGTGCTGTGATGTACGCCGGGTGGTAATCCTGAATGCTCTTACTTACACT
ACCCGATAAGCTTCCGATGGCAACGCGCAATAGCCCCAGATAAGGGTTAACCCTAGACAACTTAAACCCCCCT
CCTTTTCGTGTCTAACGTAGTGTAGAGTTTGGGGCCTCGCAGCCTTCATTAGTGCCGTATACGCAGGACTGAA
TTTGAAAATCTCGTGGA"};

    private static String s10230[] =

{"TGAAAAATACTCACTGTCGTCTGAGTACGTCTCAGTACCCAGGACTCAAGTCACACTAGTATAATCCGAGA
CATAATCCTGTTTTGCTGTAACATGAATTTCTAGGGGTAGCACTCCGACACCTTCATTCGGAGCAAAGGTTGT
CACTCCCCTCAGTGTCACACCGCGCGTCCCGATGCCCCAACGATCCCGCTTTTAAGGGTCAATGCCCGATTTT
CTCACGATGTCGATTTGCCTAATC"

,"TAGATACTAACTATCTGTCGCCGCGTGCGCTCTCGGATAACTCGAAACGAGATACCGAGCAAAATTCCGTT
TTCGTGCTGTACAATGATTCCATGGGATGTACTCCATAGTCGCTTGCCAGCCAAGCGGTTCCATGCCCCCATA
GTTCCAACCGTGCTCCGCAGGTCTCGCCATGCGAAACCCAGCTAAGGGTTAAGCCTTATTGCGCAGCTGTGCC
ATTATGTCGG"

,"GTAATACAAGCCTGATCTCGCCCGGCGCCTCTCTGCCCAGGACGAGTCAGAAAGGCATTCGATTATAATTC
TCGTTAGTCTGACAAGTACTACGGGGATCCACTCTTCGCAGCTCTGTACGACGAATGGTTCCAATGACCCTAA
GTTCCACAGTCCCTCCGAGTTCCTCCGGAGAGAACCGGAGCTTTAAGGGACCATTGCCGATTTGCAAATTGCG
TTTTGCCCCTCAG"

,"TGAAATCATACTCGTATGTACGCCTGGGGGGCTGTCGCTGGTGATAAAGGCTAACTGTATTCCCGGAACTA
TTCCATGGTCTTGTTCGCAAATCAGTTTCTGGCATGCACTCCAGCCTTTGCGGAGCAAAGGTTCACCTGCCCT
GATTTCACCGTGCCTCCGAGCCTCCAGCGACCGCCTATAAGCTAATGCCATTTCCCAGTTGGCATATTTCGCC
TCA"
```

```
,"TGAATACTACATCGTAATCGTGCGCCGGGGCGTCAGTCCGCCAGCGAGTCGCACAACAGCGAACCGTAATA
ACGAGGTGTGTCGACATAGATTTCTAGGTAGTTCCTCATAGCCCTGTTTCATGACGAGATTGTTTCATCGCCC
TTAATGTCCAACGCGCCTACCAGCGTCCCGAAGGACCCGACTTTAAGGGTAGCCAACTTTTCCCAGCTCCATT
TTGCCTCCAG"};

    private static String s10240[] =

{"CAACGGCTGGTCACGTAAAATAGCGCCACTAGTTGGGATTGTAGAAGCATGCAGAAGGTCGCCAATCATAT
GGAAGACCGAATATGCTCTCAGTGCTGGTTGCTGGAGGATGACCACTAATCTGGTCGGTACCGCCAATGAAGA
TAACTGGCAAACAAGTGCTCCGTCCCCCGGCGCTTGAAAGGGATGCTCCCCGTGCGTCGCCTCTGCTCACAGC
TATCACGTTCCGGTGCCGTACTCGGCTAGGA"

,"ACGCGGTTCTCGTCACTGAAAATTCCTATAGCCACATTGAGATTAGATGCATCGTGAGTGGCCCCAATCAT
AGGGAAGCGCAATAGTTTCTAGCCCTCGAGGTTGGTAGGGAGGTACAGTAATCTGTGAGACGGGAATGAAGGA
TAACGCCATAAAGTTCTCTGTCACTCCGCGATGGAAAGGAAGTCCGGCGGATCGCCGCCCCTCACCACTGCTA
TCCCTCCGTGCCGTTCCGGCTTAGAG"

,"CCGTCCGCTCAGAACTCTGAACAATATTAGATCATGATAACGATATCCAGGGTCCCTACAAATGTCAGAAC
GCGATTTCTGCCGCTCGGGGGTCGAAGGGTTACCGTAATCTGGCGCACACGAATGGATAGATACGTGCAATAT
GAGGTTTGAGGCTACGCCGGATGTACACGGATGCGCCGTTTGCGCACTCCTCCTACAGCAGCTACTCTTACCC
GTGGCCTGTACCGTCTTAGGA"

,"ACCGGTCCGTCCTGAAATCAGTAGTCCTAGTGGGAGTTCTAAACTGAGCATGCGGGAGTGCCCGACTCATG
ATGGGAAACGGAAAATTGTCGATTCGGGAATGGACTGAGGGATACTACTAATCGTTCGGACCCCCTAGCACGA
CAACTGTGCAAAAGGCGGTCTCTCGACTGCCGGTTGGAAAGAAGGCCCGTTAGGCCCTCCCGCGACGCCTCTG
CTGGCCGCTACCGGACAATATAA"

,"ACCGGTGCGTGCCTAAAAATCATCAGCGCCATATTGAGATTTAGGTAGCACCTCGGAGGTACCCAATCATT
TGTGATGACGATAGTTCCGGTCACAGTGTCGTAGGGAGTCACGCTAAATCTGCTCGGGCCCGAATGAAGACTA
TGGCAAAAAGTGCCTACTGGATCGCTCGGTTGAGGGAAGAGACACGCGCGTCAGGCCCTCTTTCACGAGCTGT
ATTTCGCTGGGCGCGTACGCTCATGGA"};

    private static String s10250[] =

{"GCAAGTATCTGAACATATAATATCGGTCGGTATTCCGTGGATTTTGCAATTAGAAATGTACAACCTCCTGA
TAGCATCTCCCTACGAGGGCTAGAGCCGGGCGTACCCCTACACCCTAGAATCAACTTGAGAGACCTGAGCGAA
TAGCCGAGAGTATGTTAATGCTCAGTGTAATAAGCGGAAACTCGTGATGGAGGTAGCCTTAGCATCCGTGAAC
CATTAGGAGAATGGGTGCCAGGTTGTGATGCTCATGATCAAAATAGC"

,"GGGATGAATTTACACTAAATATAGCAAGTGGGTTCGTGTGACCTTCTCTATTGAAGTACATCGTGGTAAGA
CCCCCAGATTGGCGGAGGCGCTACGGTCAACCAAAAAGTCGTGCAACCGGTTGAAGTCTAGCAATACCGATAG
TTTTATGAATGTCCTTAACTAGGCCAAATGCGAATTGAGGATAGCTATGCATACTTCTTATATATTGGGGAAA
TGGATCCCGGATCGCATAGACGCCTGTTACAAATAA"

,"GAGAATGATACTTGTACATTAATATACTGGTCTGGGTTTTCAACTGTTAGAGAACATCGGGTAAGACCTCC
CACATTGCAGGAACCTCGGACCACTAAACGCTTATAGCGCGTGAAGACTCAATAGGCGACATGTTTATAATCC
TAGTATAACTAGCGAAAGTCGTATGAGAAATGCCTTAGGATGGTAAACATTATGGGGATATGGGGTCCGGTCG
GTATGTGATTCAGTTACAAGAGACA"

,"GACGGAGTTCTTAAGAATTGATTACGTGGGCTGGGGTTTCATTTAACGAGAATCGTGGGTAAGATTCCCCA
ATTTGCGCAGAGCGCTACGTGAACCCAAACATGCTTTTAGTACCACGATGAAGATCTTTAAACGCAAAAGATA
TATGTAATCTCGCTGTAAGGACGAAAGGCGGTAGGGACGAGTCTTGGCATTCTGAAACGATAGTGGGAAAGGG
GTCCGGCTTTCGCCTCATGTTAAATCAGCA"

,"CGGGATGCTTTTCATAATAGCGCATTGGTGTCTGGGCTTTCACATTGAATAGGACACTCTGGGTAGAAACC
```

```
CCCACATTGGGCGAGAGCCCTTCGGTGCTGGTAAAGCCCTTGGTCAACGGAGTGCCAGATCTGCTATAAGCGG
AGTTTTGTAATTGCGGTTAACTAGGCAATGCGAGCTTTGAGGCACGCTTGGGCTCCGGAAATTTGGGGAAATG
GGTAGCGTGAATGGCACACGTGTACAAAATAGTCA"};

    private static String s10260[] =

{"CGTAGGCTAATTCTTATAAAGAGCTGACTATCTATCTCTTGACTTTAGCCAGTGCGAAACGAAGGCCCGAT
GTGGAACGTACTAAATTATGAGCGATTCGCACGTACAGATGTCAGCGTTATCATTCTCTGGGTGCAAACAGGT
GGATGTGTGTAGCTAACAAAGGAGTTAAATGAAAAACCAAACGATCGCGTTTCTGTGGTAACACAGATTAAAA
AGACAGATTATCTCCATGCTTCTGTGGATTCATACGAAGCTCA"

,"TCGGACTAGATCTGCTATGTGAAGACGTATCCTTCATTCTAGACTTAGTCCATGACTTCGATGAACGAAGC
GCGAGTTGGGGCGAGACAAAATAGAGGAGGGCGTCTCCACTGACACGATTCTACCGTTACTCTACTCTTGGGG
CCCACCGGGCAGGGCTTTGAGATAAAAGTGCGTTTAAAAGTGAAAACCATATTACTTGTGCGTTCCGTTTATC
AGGTTAAATATGCAGAAGTATCCATGGCTCTATGGACCTTCAGAACCGTC"

,"CTGGGCTCGTATGCTTTAAGCTGAAGGACCTTTTTTCTACCTTATTCACGCGCTGTCTATACAAGAGGGTC
CGCGCGTGGAGACTGCATAAATCACGAGTATGCCAGAACAGCATCTTACTTTATTACTCTAGACAAGTGGTCG
CAGGCTGTGGTAAAAATGGACGTAGAATGAGTAACCAACTATCGATCGTTCCTGATAAACTATAAAAGGCGAA
TTTATCCCATGCTTTTTGCTAACCATCAGAACCTG"

,"CTAGCCAGTTTCGTTGAGGGAAGTGCACTTGCTGCTTCTAGCATTGACGCAGCACTTCGTTACCGAAGCCC
GATGTGGGCGAATGCACTAAAACGAGAGTAGTCCCAGGTAACAACACTACTGAGCAGTATTCACTCTTAGGGC
AAAAACGGGGCGATGGGTTTATGAAAAAAGCCTAAAAAATAAAAACCGAACTTTTGTCGGTCCTTTATACACC
AGATCAAAAAGACGCAAATCATCGTTACCTAGCTTTGTGGACTTTACAGAGTCGGTT"

,"CTGCGACATGATCTTTCGAGCAAGACACTTCTCATCTAGCATAGCCATGACTCGATGCGAGGGCCCCGATT
GGGGAGCATGACTAAATGTGAAGCGTCCTCCGAAAGACCTTTAGATTTATCCGTGGGGCAAAACCGCTAGCGT
ACAGTTGAGATAGAAAGCTGTATAATAATGAAAAAGCAACCCTATTGTCGGTCTTTTATCCAGTTAAAATGAC
AGAATTCATCTATGCTTTTTGGCACCACGACCAGTCT"};

    private static String s10270[] =

{"GACAGGTTTGACGACGCCTGCGTGTGATGGAAGTGTGAAAGTAAGCTCAAAGGGTCACTGAAAACCCCATG
CAACCCGGAAGCCGACAAGTGTCTTGACAGAATCGGATTCAAGTTCGATTTAAGCAGAGGCTCGACGGATGGA
GCAGGCTACGCGTTATACAACCTGAGCTATAGGTGTCTTTAACGGCGGTTTTCAAACACTCCGTCGGGACTAA
CTCTGTGCTGTGCCGTGGAGCACAAGCATACTGCTTAGGTCCTTAGATCGG"

,"CGTCGGAGTTTTAGAGAAGCCTGCTGGTGTGGAAGTTAAATGTAAGCTCAGGTCCAATGAGCTTCCAAGCA
ACTGGAACGCCGCTGAGTATTCTGAGACTGAATGAGGAGTATCAAACAGATTTGGCAATGCGTGCAACCGGTA
TGGCACTGCAGGACGCATATGATTCCATGAGTCTAAGGTGCTTAAGCCTATTTACATCTTCCTCGGACTCATG
TCTGCGGTTGCGCATGCAGGACACAGCAAAACATCTCGATCCCATGTAGCG"

,"GTACGGTTTGACGAGCCCCGCGGGTGTGCAGAGTTGAAACTCGAACGCTAAGGGTCATCAGATCCCGCCAC
GCACTCGAACAGCCTGCATTTTGCAAGTACGAAGTCGAGTAATACTCGAATTAAGCCAGCATGGCACAAGTGA
TTGGACTGACGGCACGGATATCCAATTCCGAGTATAAGGTGTCATTAACACGTGTTTACCTTTTCCGTGGCAT
AACGTTCGGGGCTTGGCCTGTCAGGCACAGGCGTCATTCTTGCAGTCTTTGTAGCG"

,"GTCCAGTTAGGTCAGCGCCGCGGCTTTGGAAGTTAGATCGCAACGTAAGATGACTCAGGAACCCGCCCCCA
CGATACGGGAGCCCGAATTGTATAACTCCAGATCGAGTGTAACCTGATTTCGACCAGGCTGGCCAACCCGATT
TGAACGCGAGGCGACGCGCTATGACGAATGCTGAGACTAAAGATTGCTTCACGGCGTTTTACCTATACCGGAT
CTCGATCGCTGGTCTGGCAGCCAAATGCATTCACTCCTTATTCGCTGTATGAG"

,"GTCAAGGTTTCAGCCGGCCCACGGGTGTAAGAGTGATGGAGACGTTAAGTCCTCGGAACCCGCAGCAAAGC
GAAACGCCGCCTAGTGTGTTCTAGTCGAGACAGGTATCATTTCGATTTGTTAGACTAGCGTGCGCCAGTATTG
```

```
GCGAGCGGCAGACGATTAAACATTGCTCAATCATGGTGCGTAAGCGGATTTAAATTTCCTGGGAACCTAGTCG
CGGTTGCCTGCAGTCACAAACACTACTTCTTGGATCCATTGAGCTG"};

    private static String s10280[] =

{"TTATCCCCGCTGGCGCGGCCGGCACATATCGTCAGCAAATTTCTGCCCAAGAAGGGCGGGCGGGTTTCACC
ACGTTTGCCACGTGTTCAGATATCTGTGAACGATGCTGTTTTATCTCAGGGCAAACGAGGCACGCGTAAGCGA
TATTTCCCAAAGTGAAGGGGCTACAATCACTGTATCTGACAGTAGGTTTGTACGCGGCGTCGACTTTATTGAG
AATTAGTCGGCGAAGCTCGAAAGGGAAGCGCACGCTGGAGCTATGAGGCACTTGATACAGT"

,"TTCCTTCCTGCAGGCGGCCGCGCACATTGGGTCCTAAAATTCCTTTCGCCAAAACGGGCCACGCAGTAGTC
ACACCGTTGCCAGCTTCACATTTCTTAACATGCTCTAGTAGCAGCGAACGAGGTCACTGTAGAAAAATCTACC
GAAAGCACAGGGGTTCCATACATCGTAATACACGTGTTCATGAACCGACAAGCCTACATTGATTTGAGCCATG
TAGTGAGGAGCTAGCGTGAAGGAGGGGAACGCGAGGCCATAATCAAGCACTTATAGGAT"

,"ATTCCCCTCCCCGCCGCGTATCAATTGACTTCAACAAATTCCTTACCCAAAGAAGGGGGCGCAGGCTTCAC
CACGTGTGCACCGTTTACAGTTATATAACCGCTTTGGATAGGGCTACGGAGCCCTCCGATGAAAAATCTTCCA
CGATAGAAGGGTTTTCTAACTCGTCCGAGGGATTGTAGAACACGCAGCGTTGCAAGTCGCCATTGAGCATACT
GCTACGGCAGAGTGAAACAGGACTCGCCATGAAAGCCAGAATCCTTCAATCAGCTTATATCATG"

,"GTCTCCAGCTGTGCGGCCCCGCCACAAATGCCCACAATTTCCTTCCCAAAGAGCTGGGCCGCAGAGGTTTT
CCCACGTGTCAAGCCTCACCATATCTTAACATGAGCTTAGCTTCAAGGGAAGAGGCACATCGTAAAAATACTT
CCCACGTAGAAGGGGTTCTATAAATCTACGATGCAGCATGGTCTTGACAACCACGCGTCGTTAGCATTGTAGG
AAATGTATCAGGGCAGAGTGCAAGGAGGACGACGCAGCCCCTATCTGCGACTTATCG"

,"TTTCCCCCGTGCCGGCCCCGGCGACTGGCCTCGCAAAATTCCTCACCACAATTCGGGCCGGGAGAGTTGCC
ACCTACGTTTGACCGCTATTCAGAATTTATACCTTGCTTCAATAGTTCGGAACAGCAGAGCTAGTATCAATGT
GCCACAGATAAGGGGTTCAAACACGTATCTCAGCTACGAGTTCGTAAAGCGGCGCGTCTTGCCTTTGACGAAT
TAGTACCAGGACGAAACAAGAGGTCGCACGAGACCAATATTAGGCATCGAACCTCT"};

    private static String s10290[] =

{"TTGCATATCGCGGACTGTAGTCTGCTCCATGCAACCGAGCCTAAGATCGTTCCCCGGTGTATGCAGCCAGT
GATGAGGGAATGTTAACGACTGTGATGAGAAACGTGATATCAATTATCACCCATATTCGCTGAAGCCCTGCCT
AAGGAGCCCCCGTGATAGCACTACGTGACGCGAGGCAGAGACCGTGCCTAACGCAGAGCTGCATGAAGGTGTC
AGAGTGAATCGTGGGCTGCGATTTGCTACTCGGGCTAAACCGTGTCGGGCGCCCGACTTAAGTTAAA"

,"CTACTAAATACGACGCTGGGGTAGTTTCTGCCATCCGTATCCGAGTCCTAACTGTTCCGCCCGATGCCGCC
CGACGCCGGAGGGCTAAGTTATAGCGCAGAAAGACGTGAATATCATTTACAACCCATATCGGCATACACCTTG
ACATCAATCGAGCCCCGGTTAAAACCTAGACCAGCGGGCGGTGACAGAGGCGGTGCCGAACGTAGCGCTGCCT
GTAGTGGTTGTTACAATCGGATGGCCTGGAAAGTGATTCATCGCGCATTACACAATCTGCGCTGCCCCGTCGA
AAGTATA"

,"TTGCATAGCGAGATCTGATCGATGCCGTCACCTGTACTAGCCAAGCTCAAGAAGTTGCCCGCGTTAGAGGT
CTCTACGCTGGCTGAGGTAAGTAGTTACAAGCTGAGGAAAAAGCGTATAACTATACACCCATTACGACTATAC
GTGTCCCAGGCCCCAGTGCTATAGACTAGGCAAACGGGCGAAGCAACGCGGTTCCAAGCGAAGGTCTGATGGT
TGTTAATGCAATGGGAGTCCGGAAGTTGATCGCCGGCATAACCTCGTCCGCTGGCCCCTAGTCAAAGTAGAA"

,"TCCATAGTCGCGTGCATCGAGTCTAGCCAACCGCTAGTCGAGCGTCCTGATAATACTTTCCGCCTTACTGC
CACACCTGGCTTGAGGATCGTAGTTATCACAGTGGAAGAATAACGCTGAATTATTATACCACCTAGATGCGTT
AACCCTTGCCATAGAGCCCCGTTGGTACGATAACATTCAACGGGCAGAGCTAGAGAGCGTCCAGACCGTAACA
CTCCGTAGGTTCTCAATAATCGAGGTCCGGAAATGTCTATCCGTGCATGACACATGCTCGGGCCGGCACAACT
AAATGTCATCAA"

,"TTCGATAAGTCCTGGATCGATCTTCTCCACTCGTCTATCAGAGCCATACCGTTCCCGTGTCTGAATAGCAA
CCACGTCGGAACGATGATTACAAGTGGGAGTGAAGCAGTGGAACTGCTTTTACCCCATTACCGTAACTCTGCC
```

```
GTAAGGCCCGTGTATGAACCTCAGTACAACGGAGAGTGACAGCGTCCGAACGCGCTAGCTCGATGGTGTCATA
TGAATTGAGGGCGCGGAGAGTATCATCTCGTGCAACACATCATGCGCGGGCCCGTTAAATATAA"};

    private static String s10300[] =

{"GTCATCGCTTATAAAAGGCGGGTGATTCGTTCAGGTAGCTACCGCCCGCTCGTGAGCCGATCCCGACTATC
ATATAAAAGGTGCCCCTAACAATCTTCACCACCGTGACAACTGGTACTAAAGATTCAACTGGTTCTATGCCCC
CTAGCATTGAGTTTGGCGGATGGATCCTAGTAGACTCGGACGGCTGGTACTGCTTGGACCGACCCGGTATATT
GTTAACACAAACTGTGGCCATGACCATGCCGCTTCGGCACGCCCCCCAAATAAAGTCACCTTAGGCTAAACGA
CAACCCCAGAC"

,"GTTCCGATCTACTAATAAGGGTCGACTGAATTCATGCAATGGTTCCCCGGTTACGTAGTATGCATACCAGA
TTACAATTAAATTGCGCTCTTAAAAACTTTCCAGTAACTGTGTGACTACTATACTAACGTGTGTTCCTAACCC
CCCTGACATTTATGTGAGATAGACTAAGTGTATTGAGCAGCAGGGCTGGTAACCGTTTGGCAGCCGGTTATAT
TTGAGAACCCAAAATGGTGCGATCCACTTAACGGTGCTTGGAGCCCCCGGATAGTGCATCGTCAGCGCACGGG
CCGAAACGCTCCGAAACT"

,"GCCCTACTTCAAAGAGGCAGTGGAATTTCATGCGATGCATCGCTCATGAGCAGCTACCGTACATCATGAAA
GGGTCGCTCTCAAATCTTGACGACGTCTATTTTGATAAGATCTACATGTTCGCTCTAGCCCCCCTGACTTAGT
GTAGGTAGGTACCAAGGTAAGCAACAGGGCTGGTACAGTGATGTTTCGCGGTCATATTTTGAAAGCCAAAAAT
GTCACTAGCCTTGGGTATACTGGGCAGCCCCGAAATGATTCATCCTAGGCTCGACGCAAAGCCCCCGAAC"

,"GTTCCTTACTTCTAAAAGGGCGAGTTGATTCATGCATGGCATCCCCGTCAGGTACGCTGCAACGCATATGA
CATTAAAAGGTCGCCCTTACAAAAATCGTACTAGGCATGCTTGACACAGGATACTTCTGGTCCTAGCCCCTCT
CGGACTATAGTTTGAAGGGAGATTCTCGACTATGAAGCGGAATGGGCTGGATACCTCGTTTTGGCAGACCGGT
TGATTTTTGAAGCCCAGGAACCTGTTGCACATGACCTCCGGTCTGTTGCTGCCCTCCCAGTCGTATTAGGACG
AACCGCGAGATGCCCCGAAC"

,"TTCTCATCGTCTTAGAATGGGCGATGCGAGTGCTTGCCTGGCCTCGGTCATGGAGCACTAACGATACTTCA
ATTAATAATTGCCTCGCAAAAACTTTCAGTAACAAACTAGTTACAGATCCAATCGTATCCAGCCCCTCGCAAC
TTGATGAAGGTAGGATCCCGGTATGAGCCGCGGGTGTGAGCCTCCGGCTAGCCCGTGTAATTTTTAAGACCCA
AAATCGTGCACGTACCTTGGTTGCTGTGCGACTCCGATGTTAAGTCACGACGGCTGCAGCCAACCGGACA"};

    private static String s10310[] =

{"CTCCGTACGTATAGGTTACTAAATGTCTATTCGATATAATCTCTCATCCTGACGGTCGGTCTGGAAACGCA
AGCCACATTCGGAATGCCCATCCACTTGTCAACTCAGTGGTCCGAAGTCGAACTGACACTATTACCGTCTTCG
TTCTCTCGCGCTGTATAAATAAATGCGACTTAGATGTTTACGTCAAGCAACAACATGTATGTTCCGAGTGTTA
ACACTTGGGTGGGTGGAGCGGTCTTCGAAGGAGGCAAGGCTGGTGCAGCGAAAGGAGTACGGCGCAATGATTA
CTAACTACAAACATA"

,"TCCGTCACGCATGGGTTTCTCGTCAAACGTACTCGTGTCGCTATACTCATCTGGCAGTGCAGCGTCTCGGA
ATAAGCAGCAATCTCCGACATGCCATCCGGGTTGTCGTACGCTCATGTGGTGCCGTAAATTGAATCAAACCCA
GTTTCATCGCCTCTCGTACTTGCGGTATACTAATAGCGGAGTGCTACACTGTTGTGCGCAAACGATAAACGTT
AGCTGTCCAGGGTTCTTACCTTCGCTGGAGCTTCTACGCAGAGGGACACGTGGTGGGATAAAAAGAGATCGCA
GATTGATGACTGAATAGCACATA"

,"TCGCTACCGATAGGTGTACATAAACCTCGTCGTGCATATCACTTCCCTTCCTGGCGTCCGGACGTGAATAG
ACAAGACATTCTCATTGACCTCCGCTCGTCCTGTAAGTCCATGGGTGCCGAAAGTCAGATAGACACCCAATTC
TACATGCCCTTCTTGCCTGGATTACAAATAGTCAGGGACATAGTGTTACGGCGGCTGCCGTTTTCGTTCCAAG
GGTGCCATACCTGTTGCGTGACGGTTCTCCGATGGATGCGAGGTTTGGGTCCGCTAAGAGAGATCGCGATGTA
TACTGAATGCAAACAAA"

,"TCGCCTCATAATAGGGTTACAGAACGTTCCGGTTGCCATACTCCCAATTCCGTGCGGGCGGTTGAGACATA
GACACACACTATCATAATCGCCCAGTCTCGTTCGTCAGGTCTAGTGGTGCGCCATCAGCAGTCAATAATTACT
GTCCCTTGATATCGCCCGCGGTATTACATAATGGTCGATGGAAATAATGTGTATCGGCAAGCAATAAAACGTT
```

```
ACCATCTTCCGGGTTCTTACGTTGCGCTGGACGTCATCGAGATGGCGACTGGTGCCATAAGAACGAATTGCAA
TTGATATCGAACTAGAACAAGT"

,"TCGCTCACGCATAAAGTTTACCCTAAATTTCGCGCGGTGCGTCTAACTCCGTTCCGTCGTCCGGCCTGGAA
TAAAACAAACAAACTTCCGAAAAACCCATTGCTTTGCCAGTTGTGTGCCGAAACTGATACGCAACAACTTCAG
TCTGCCTTGCTTAGCCGGATCTATTGATGCGTACGGTAAAGTTGCTCCGTCGAAGCTATACAACACTGTACGT
TACAGATGTGCTACTCGTGGCAGATGTTCATCAGGGCGGTAGTTGAGGCACTAATATGATGATCGGAACGTAT
TATACGAACCTACCCCAGA"};

    private static String s10320[] =

{"AGAAAACGCCTTCCTTGCTCTTCGAATCGTTTTGCCAGTTGGGCCGTTGGGACTTCGTCCGAGGAAAGAAA
GAGTAAGCACAGTAGCCCGATCCGCATTGCACATGTAATCACAGTCGTCAGTGTCTCAGAATGGGCACTCCCT
TTGTCGCAGTAGATCTTATATTCCAATCGTCCCGGAATCGTTTTTACCTCTTTGGAATACCAATACCAACTCT
GGCACCTCCTACCTCTTCATAGACGCTCCAGGAGGATCGGGCAGTTCGTAGTGAGGGTGCAGCGCGTGCCACA
TCACGTGATTAAGGGTCGGGTCCAGTTA"

,"AGGTAACGCCCTCTGCTGTTGAATATAGTCCATGGTATGCCGTTGGCATTCGTACGAGGATAAGACAAGGA
ACTGATATGCACGTCGCTGTAGCTAACGTTACTAGCTAGCAGTTTTTGCAAAGAGGCAGCACCATCTCCGGGT
AGTCTGTTGACTTTTGGCCCCAATATTTGCATCCCCGTAGATCGCAATCCAACCTGGGCACCCTGCTACAGGC
TACAGTAAAGCTGAGGGGAGATCATGGCGATACCTAGTAAGGGGGGTAGCAGCGGTCGCTCATCTCAAATGTT
AAAGGAGTGAGCTGAAAGTTAT"

,"AGGCTAGCGACCTCCTCTTTCCTATATGCTATTCCGATATCCATGTGGACATCAGTACGCAGGTAAGAGAC
TCAACCGATGACGATCGCAATCGGCCACGTGTAATACTGAACATCTTATTCAGAATGGCACCTCCTTTCCGCA
GTATGTGCTTTATGTAGTTCCGCAACATTTGTACCCCGGAGCATCGCAATCCTAGTCTGGCCGTATCTACAGG
TCTCGATATCTACGAGATCGGTAGGTCCTAGTGGGGGTAGCTGCGGCTGTCCAGACCATATTCAATAGGGGTG
CGTCCAGTTTA"

,"AGAGCTAAAGCGCTCTGAGCGTTTCTCAATGTCTTATCGTACGAGTTAGTTTTCGGATTCGAATCGAGGAA
AGAATGAGCTAAACTCGAAAGCCAGCTCGTATAGCCTACTATAGTAACTGGCTAAGTTGCTTCAAAAGGGTGA
CCCTCCGTTTGCAAAGTTCTTTAATTGAACAGTGTCCCGGGAACTATGTATCCCGGTGGCATAACGCAATCAA
CCTTGCGGCCTGCAGCTCGACTCGTAAAAGTGCCGGAGTGAACTCCGTAGTCCTTAGTGAGGGTTGACGGCGT
CCCTGCCCTGATGGTAGGGTGCCGGCGTCCATGTTA"

,"GAGTCAACGCCTCCTCCAGTCTCAATTAGTCACTTCGTGAAGTATGCCTGTTTGGGCATTCGATACGCAGG
AAGAGAAAGCATAACCATAAGACTGCCGTAATGCACTAAATGAAATCGTATGCATGTTTTCAAAGATGCGGCC
ACCCTTGCGTTGTCTTTAATTGTAGCGTTCACCGGATAATTGACTCCCTGGTGACATGACTACCAACCTGTGC
GCCTGTCGTAGCTTGCTTAAGGTCGACGGGAGAGTCAGGCGATGTCCCGTAAGGGGATGGACAAGCGGCTCTC
ACAGTAGTAAAGGCTGGCCGCTCGCAGGTTTC"};

    private static String s10330[] =

{"GCTCTGGGTTCTTGGTGGTACTTCTGAAGGCGGCCATACAAGTGTCCAAGTGCGTTACTTAAGCGAGATGA
AAGGTTAGCCGAAAAACGGCCATCGTAAGCTCTCAGATTCGAAGCGCGTAAAATTTTGATATCTTGGCAAGAG
ATACGCTTACACGACGCCGTGTACCGAAGGCCATAACTACACCAGGTTGGCGCTAGCAAACTCACACCAGGCC
TACCCGCAAACTTTCAAAACAGCCGCAGAGTCGTGTGCAGTTCTGTTTAGGCTCGTGGTCTCAGGCATCGGAA
TGAACTCGAATCCTCAAAGATCACTTCCACATGCGCTTCTCAAACCTTTA"

,"GACTTGGGGGTTACTGGGGGCTTACTTTCGAAGCGCCATAAAAGGTTTCATGGCGCTTTCTACCGGGACGT
AAAGGCTTGCTAAGAATGCGCAGTCGGTAGACTCTAGATGAACCTCTAAATATTTGTATCGCGGAGTACGTTC
TACAGACGCCTGGATCCGAGCCGAACCTCCCAATTTGGCCGTTGCCAAACAACACCTCGGGGATCGCGGGAAC
ATGTCAACAGACCGGAGACCAGGCTAGTACTGGTCTCGTTCGCTTGCGTGCTTCGGATCAGGTACTCAAAAGT
CCTGATAACCTTCCGATGTAGCCTTTAGACTCCTAA"

,"GCCTCTTGGTGTATCGTGGGAGGAGTACTTTTGAAGCCGCCCATGCAAATGCTTACTGCTGTTCTGCAAGG
```

```
AGGAGAAGGCGTCGGGAAAATAACGCGCATCGCGATAGTCGCATGAGTGACACCTAAATATGTTGAGTATTTT
CCAGGGTTACGGCTTCTAACGAGCCCATAGTCGCAGGACTCAACTACCACACAGTGATGGCTCACAAGCCTAA
AAAGGGCGATGTCCCGATCCCCCAAACAGGCCGGAGAATGACTAGTGTCCTGACTTGCTTTTGAGCGGGGTCG
TTCGGGCGTTGCGTAACTCGATCTCTTATCCAATTCACTGTGCTCCTAAAACCTTTA"

,"GCATATGTGGGTACTGGTGGTACTTCCTGAAGGCGGCCATCAAATTTCTAGGCGTTCTAGCGAGATTATAA
GCCGTGCGCAAAAAACTCCAGAGTCGGTCAGCCTTAGTGTAACCCTTAAATATTGTACTTTGCAGAGATACGT
TCAGCACCGCGTGAACCACCGATGCCTCACAACCCCAGTTGGGTCTGTGACACCAAACAATGAGATACTCCTG
AACTTTTCAAAAACTCCCCGGAAACGAGGTGCTGATACTGCTTGTTTGCGCCAGGTGCTTCGCCCTGGTACAT
CGGTTCCTCAGACCATGTCCTATGGCTTCTAACCCTTTA"

,"GCATGGGAACTTGAGGGCTTCTTAGGGGCCCATACACATGGTTCCATGGCACACTAACCGGAGTAGTAAGG
CTTGCAGAACAAAAGCCGGAGTAGGAGAACTCTAAGGTTGACCCAATTATTTGTATGCCTGGCGAGGTTCTTC
CTTAACGACCCGCCTGGTTTCCAGAGCTTCACTCCACCACTGGTTTAGCGACACAACACCCCGCGATGCACTA
ACATTCGAACACCGGATAGCGGGCAGCATGTGTGATTGGTCGCGGGCGTCTGGCCGAGTAAGCTGGACTCTGG
TACATTTCATCTGACCCCTTTAACATCTAA"};

    private static String s10340[] =

{"ACTAAGCTTCTTACACCGAAGTTGTTGCGTACTTGGTATTCTGCGTTACATATCTATAGGGACAAGGGCGC
CCTTCAGTCTTCTGCGCATTCTTCGTAAGTTCGGGCCCCGATGTCATCTACTCAAAACAGTCGGTGGTATCGA
GCCTCACAGTGAGAGATCTGATCTTTGTGATCGTCCTACGTTAGATTAACGCTTCGTAGCGGGACATTGTAAA
CTGCAAATCCGGTCCCCGGTCGAAATTGATAGCCCTTATCACCGTCAAGGTCAATATCCCGACGCGAGGCGAC
ACACTTAGGCGAGATCAGATGATCCTACAGCAAGGG"

,"ATCAGATCTTCTAACCAAGACCTTGGATGACGGATTTGAGGGTTCGCTTACATATCCAAAGCAGCGACGAC
CCCTTGTCTCCTGCTTCCTCAAGTTGGGCCCCTGTATGACCTATCTCACTATTCAACGTTCGCGACTCACCGG
TACAGGACATGATACTTAGGTCGTCCATCCTTTAAAGTTGACAGTCTCGTCGATTGGACCGCATGTCAAAAAA
CCTCCAATAATCGGTTCCCCGTTTTCAACTTCTGACCGACTAACATTGTCTAAAGTACATCTCACAGCGACGC
GGAGGGCGACGCTAATAGGGCGAGTACGAAATGTAACACTACTATTCGAGGG"

,"CACGGATTTCCTCAACACAGCAATGGTTGATTGATTTGGAGCTCCCTACATATTCCAAGCGCAGAGCTGGA
TCCCCCCGGTTCTCGCTCCGATATCCTAAAGTTCGCCCTCGTATGCCTTCTTCACTAGCGTCGTTGTCTGACC
TACCATGACAGTACGATCTGCAGGTCCGCCTCATACTCAAGGATACCGTTCCGATACTAAGGGAGAGTAAAAA
CCCCCAAAATGCGGTCAACAGTTCTCTAAATCTCTGACATACTAACTTGCTTAAAACTAAATTACTCACCGCG
AATACCATAAGATGGGCGAGTTGAGATTGAACTTTTTTAAGCGCG"

,"ATTCGAGTCTGTATGACACAAAGCCATGGTTTGAGCGAATATTAGAGATCCTCGCAAATCAGATCCTAACG
CACGTTTCGGCCCTCAGTTCCTCGTTCAGTTCTCTCAAGTTGGGCCCGTATGCTCCTTCACCACTCATCAGTG
GTCGCACTCCCGCGGACACCAACTGAACTTTCGAGTGCCTCCAATCCTTAGATTGAGCGTCCTAACTGGACGC
GTTAAACACCCAAAAACGCGGGTCCCTTTCTTCAAATTCTGGACCGCCATAATCATGCTTAAGTCACACTTCC
TGCACGCGGAGCGGGCACACAATAGAGGGTAGGTACAATTCGAACTGATTCTGACTGAGCG"

,"ACTCAGATCTCTTCTACTGACATGGCTGACCGATTGGAGTCTGGCCTTCAGAATTCTAAGGCCCGAGCGGC
CCTACTTCTCGCTCAGATCTTCTTTCAATGTGGGCCTCCAAGACTCCTATCCACATTCCACTGGTGTCTGAAC
GATCTGAGTACACGGACTTTGATATTCGGAGATCGCTCAACAATTCTAGAGTTAACCACTAGTACAGGAGCAT
GTGAAAACCCAAAATCGTTCCCATTTTCAAAATTTTCTAGCACTCCTAAGCTATGCTTAAAGCTCAACATTCA
TAGGCACGTGTAACGGCTCCCTAATAGAGTGGACGGAAGAAGTAGCTATATCTTAGGTCTGACAT"};

    private static String s10350[] =

{"GTTATCGCATTCTCACAGTTACTTCACCAGCTTCGCACTTGGGACCATGCCAACAAGTTATGATCCAGCGA
AAGCTCATGAAACATCCAATTCGCGAGTACGATAAATGAGGGACGACTCCTGATCTCTCGGGTAAATGCGGAT
GGAATCCCATTCCCCTTACATTACGTGCGTTAAAGACGCGAACGGAGCTAACTCGCAAAACATGGGCGGTCTC
TACGCCGGCTTACGATTTGAAAACCTTAGGGCTTCCATTCGAGGAGTCACATCTCCACACATAGGTGCACGTC
CATGTCGCAATTAAGCGTATTGGATGATCGGTATCCTCACACTTCACGGGTGGTTC"
```

```
,"GTTTAGCATTCTTCACCTAGTCTACGGAACTTCCGTAACAGGGACATCGCAATACAACGTATTGACCTACG
AAGCTCAATATAACCAACCTGGAGTAGCGCTAAATGGGAGAAGTACCCTTACTCCTCGGGTTACGAGGTGAGC
CCGTCCTTAACCTAATAGGTTGGCGTATGGCGGGAAGCAGGCTAATCCGCGAAACTGCAGGTCGTGCTGACCG
GCTTACATTTAACAACTACCGAGTGTATTCTATTGCAGCTTCCAAACAATTCGTGCAGTCGAGTTAGCAATAG
TACGTAGTTGATAGCTTTGTTCCCACTTACAGGGACTCCT"

,"TGTTCTGATCTCTATTATCTTTACGGAATCCGATACTAGGAGCCATTTAAGAAACTATTCTTCGCAAGGAC
GCAGAAATAACCTCAAGCTTGGGAATATGGGACTAAATGCAGGGAGGAACTTTAACTTTCCCGGTCAATCGGT
GGACCCTATTAACATTTCCCTGTAAAGCGTGCGTGTCTAGACGCGAAGCGCGATCCGGCAACACTGGCGTCGT
GCAATCACGGCTCAGCTTTGGTAACACCTTCAGGAGTGCCATTATATTGCTTCCACGCAACATCAGGACGCCT
CCAGTTACCCTAATTAACAGTATGTCAAGGTCTTGTCCAATCACTTATCGGTGCATTCT"

,"GTTACTCGCATTCTCTACTTTTATCTGACAGGACTTCCAGCATACTTGACGAGCCACGATAACAAGATTTT
GACCTCACGGACCTTCATGCTAACCCAAAGTTGGGATAGGCGATATAGCAGGGAAGAATCCTTGGAACCTTCG
CGGTTAAATTCGTGGGGAACCTAAATATCCCTTTATAGACTAGGTATTCGCAGCGACAGGGCTTAAACGCAAA
CCTGCGGTCGTCTCACGGCCGTTACGACTTGTACACACTTCACGACTGGCCGTCATTTGCTCTTCCACACATC
ATGCCACGTGCGCAATGTAGCGAATAAAGTCTATGCATGGCGTTAGTATCCTCCAATTTTATCATTGGTATAC
"

,"GTTAGTCGTACTCATCTTTAATCCTCAGAGTTCCGATACTTAGGGACCAATGCATACTATAATAACGTCAG
GGAGCTCAATTCACTAAGACGTGAGTCGGGCTACAATGATGAAGAGTCCTTTACATCCTCGGTGTACAGATGC
GGTTAACCCTTTACTTTCAATTAGTGGGCGTGATTAGACGCTGTAGGGGTCAACCGCAAGCCGGGCGTTCGCC
AGCGCGGCATAACAGTTTGAATCAACTCACAGGACGTCCAATTGCATGGTACCTTCACTACTTAGCACTGCCA
CGTGACGCAATCAAACGTAGTTGGTGGAGATTTGGTACCCACACTTTCGGGTTCT"};

    private static String s10360[] =

{"CTGGACTTCCTCACGCCGTGGTCACCTGTCCTGCGGTTTGGGCCCAGGCCACAAGCCCCTAAAGGTTTCTC
ATACAGACTTGCCCTTCATTGACTATGGACCGATGACTTCGAATTCTTTGACGCACTAAAGCGATAGCACTCT
CTGCTCAGCGTGTAGGTGGGCTCCAAATGATGAGGACGGTCTGCCATGGCAGGAACACTCTTGGAACGACAAT
GCGCCGTTCCATGGTAGCAAAATTAGAATCAGTTGATTGTCCTGCGGCATCACGACTGCGACCCGAAAGGGGT
AATATATTCAAGTCCACGCCAGCCGGAAGAACCGCGTGGGAGACGAAAAAGGACACACTAACG"

,"CGTGACTTCCTTCACCTCGCGGTGGCCATTGTCCGGCTTTGCGACCATCGACTAGCCCGATAAAGGATGTT
CTACGAAGGGTGCTCCCTAGATCGCGTAGTCGCGAGTCGACGTGGAAATGCTTGTGGACGCACCTTCAAACAG
GAATCGACATTTCTCGGCTCGACGGGGTAGTTGGTGTCCATAATTAACGGACGGTTGGTGATGAGAAGCCACC
GTGTGCAATGAGATCGATTCGGCTGCAGATAAAAGGATAAACTAGTTGTTGTTTATGCGGGCAATCGATGGGA
ATCCCGAACAGGGGGTAAATTTCTTAGTAACCACCTCACCATGGAACAATCCTGGCATGGAGGATAGAAAGGC
ACCAATTCTTGC"

,"CGTAGTTCTCTCCTAACTCTGCGGCCTGCCTGTTTCCTGCCGTTTGCCGGCGCAAGGGAATTCACGCTATA
AAGATGGTTCTCATAAAGGTGCGTCCTACTTTAGATTCGTGAGCGAGTGCAGTAGAATCCGTTAGAGGCACCC
TCTAACATGAATCAGCAAGTGTCTGGCTCGACGCGGTGAGTGGTCACATAGTGGGGCAGTAAGCTCAAGGAGA
AACCCGTGTAGCCACAAGCGATCGACCCGCGCAATACGTCACGACAAACTAGTATAGTTTTTAGGGACTACGC
TCGGATGCGAAAGGAGTTAGAATTCTATCAAGCCCCGCACCGGAGAAAACCTGGCGGAATGAAAAGGAACCAC
TGTACG"

,"CGTAGATCATTTCACCTGGTGGCTGTCCGCCGTTGCGTACCCAAGGCGATCGCCCGCATACTAGTTTCTCC
AAGAGTCGCTCCGTTCTACTAGTGAGGACAGTAGCGTAGGTACCTTTTTGGTGGACTCTCTAAACGAACCCGA
GTTGTTCGCAGGGGTATGGGGTCCATAACGTGGAGAGGGTTTGCCAGCTGAGAGTACCCGTTGAGCTAGAGAA
GCCGCACCTGGTGCCGATAAATGAAGTACTACGTGTAGTCGTATTGGGGGCATAGCTGCTGATAACAGCCCGG
TGTAAACGTCTAGTAAAAGCTGCCTCCGGAAGAAATCAGCGTCGGAAGAAAGGCACCTCGT"

,"TGAGGTCTTCGTCCCGTGCGTGCCCTGTTCTGCCTGGACGTTGCGGCCCACGGCGACACATAAGATGTCCT
ACAAATGGCTCCAATTTTAGTTAGAGTACGCGTGACTGTGTGAATCTTTTGGAACGACACCTACAAGGAACGT
```

```
ACTGTGCTGACTGACGGGTGAGTGGGACATAGTAGAGACCGTGCCTAGTGAGAAGTCACTGGCTGACAAGAGA
GTTCCTACCAGCGCGGGAATAATTGGACATACCGTATTCCAGGTTTTTTGGCGCATAGTCGGGGTTCGAAAGG
CTGATAATTCTTATGAACTGCCCCACCGAAAGAAAATGCCTGCGGAGAATAGAAAAGGACCAATTTACC"};

    private static String s10370[] =

{"GCCCTGCAAGACCCCATACAATATAACGTCCACAATCACGTCCTCGGCCTAGAAAGCCATGGCAATGGTGG
TTTGCGGGACGTGCTGATATTCTCAACCTGATGATTTTGTCATGGGTGCGTATTCAGTGCGACGCAGCCTATC
GGGCACTTAGTGGTCCCCTGGCAGGACCATTTGCCTAAGGGAGTTCCGGCAACATATGCACATAAGTGGAGCT
AGTAATACGGTCTGATGAGGCTTGTAATGTCGTCTGCGAGCATGAAAATATGAGGAAAGTTAATGTTCTATCT
CTCAAAGTCGACGCGAATGCCAGCGCTAAGTAGACGAGTCTCTGTAAACACTTGTGCCAGTAGTGTATGTCGT
TACAACAAG"

,"GCACGTACCGCCCCTCACCAATAACCCCCATTATTTTACGCCCGGCCTAGAGAATGGCACGAGGTTTACGA
AGCTTTATATCTTCCAAGTCGAAGAAGTATTCTCATAGGGGTCTGATGGATGCACAGGAGGCCTTGGCTGAAC
GAGGGTTCCTCAGCATGCAGGCGAATTTGCCGAAGGGGTACTGGCAAATAACGTGAATAGTTTGGGACGATGT
AAACTGGTACTATGATCTGATACCTTGTCCTGGGAGCGAATAATATTAGGAATTAAATCCGTACTCAAAGCTG
CACGCACGGATGGGCCCGGCATAGTTGGCAGTCTCTTCACCAACTACTCTCGAGATTGTCTTATCCAGTCGAC
AAAG"

,"GCTGCACGCACCCCTAGCCATAAAACCTTTCATTATTAATCCCCGGCCATGAAAGATGGGAACGGGGCGTT
TCAGGCGGCGATATAACTCCCTCGAGTAGTATTCCTCTTGAGTGTTCTATCAGCAACTAGGGCTATCGATCGA
TTTGAGGGGTCCCCATGGCTGGCCACATTGCCTATAAGGGTTCGGCAATATAGCCTAATTGTCTGTGAGCATT
GAAATCGATCAAGCCGTTGAAACTGCTCTTGGCGAAGCAGTAAAGATATTTGAGGCACTGATTGTCCTGCATC
CAAGTAGCAGACGCAGAACGCCCGGACATGTAGCGACTTCTCCTTGCAACTCGTCTCAGGGGTATGTTGTCTG
AAGTCAGACAAG"

,"ACCTGACGCACCCCACTAGCATTAACACTCGTAATTAACTGCCCGGTCACCTGAAAGAAGTAGCAAGCGGT
CCAGTTTCCCGGGCCCGCAATACGCCACTGAGTATGATCCTATAGGGTGTATACGATGACCCATGGCCTTACC
GTGCGAATTTAGGTCCCCCTAACCCCGGAACCTATTCGCTAAGGATTCGAGGAACATACGGCCTATCTTTCGG
ACCATTAGATCGTTCTAAGCCTTGTAATCTGTCTGAGGAGGGATAAAAAGTTGAGGTATGATAGTTCACGACT
CTAATAGATGTATCCGATAGCTGCCCGGCATAGTAGGGCAGGTTCTTCAACACCTTGTGCAGGCTAATGTGAT
ATCAGTCCGATAAG"

,"GCCTGCAAGTACCCCTACCAATAACACTACAGTAATCATCCCCCGGCCATGAAGAATCGGAAACGGTGGCG
TTTGTCCGGCGGCTATACTCAGTAGGTAGTATTGTCAATGGGGTCTCACTAATAAGACCGAGGCCTATCGTTC
GATTGAGTCCCCTATCGCGAGACACTTTGCCCTAAAAGGTATCGGCAGAATACCCATAGTTTGAGCGGAATAT
AGCGCTTAATAGCCTTGTAATTGTTCCTGAGGCACGGATAAAATATAGTGTGTGAATGTAATCTCGTATCCAT
AAGTACGACCGGATAGTTCGTACAGGATAGATGCGAATCGCCTATCAAACTCAGTTTTGCATGTATGGTATTT
GCCGTTGCAAGAACA"};

    private static String s10380[] =

{"AGTAAACTATGGAGGCAACCTCAAATATGTGATTATAAGCGCGATCGGATATCGCACTGCTGCATAACCGG
GGACTTGTGTGTACTAGGGCGTGAAGATCTTTAGCAGAGCAATGCCATGTAACGCGTGGTAGTGTTCCATTAT
ATCTCGACTGCCTCACCGGGACGCGCTGCGCGACGGAACGTGGCTTAGGTTCTCGGGGAGACTGACTTTCTCG
TTAATGCGCTTAATAGTTTTGCGCCGGTCCGCAGATCTCTATACGTATCCTTCTGAGCGGCGAGCCGGCTTGG
CCACAAAAAGTTACTTCTCGGCTCTATGCTTTAAGTTAGTGTGAATACTACCACAGGCACCATGTGTGAACAA
GAGGGCCTGGCAGGTGTTCTAA"

,"AGTCAACCATTGGGACCTCCTTAATACGTATATAAGCGGCTGGCAGAGCGCCACTGTCAGACAAACAGGCG
CATGATTGTGTCATGGGTGTGATCTTTTGAACCGAGGGTACCTATAAGGCGTATTTGTCTAATCGATTCTCGC
ACCGGCCTTACCCAGGAGTGTGCATCCGTGCGATCGGTGGCAGTTGCCTTACGCGGACATTGATCTCTATACC
TAAAGGACTATTTTCTCGCCTTCGCCGCGAATCCTCTTCTAATTAGTAGCGGCGCACCGTCTGTGCCCAGCAG
AAAGTTCTAATGCGCGGTCTCGTATACTTACGGGTTGATCAATACAAGCGGACAGTTGCGTAACGAACACGGG
CCGCCAAGGTTTGAAA"
```

,"ACAACCCTTTAGGGCCTCGCTTCCAATCTGGATACTTAGACCCGTTGTGCAAGCGACCTGCTAGCATGAGC
GACATGGTGTATCATGGGACGCAGGTCTTTGTAGCGAGCGACGATCGATAAAGCTGGTATCTCTCATCGTACT
CGACGCCCTCTATCGGGAGGATGTGGCAACGTGTCAGTACGTGGGCTATGTCTTAGCGCGCATGGGATCTCAT
TAGATTAAAGTGGTCATTTCGCCTGCCGAGATCCTCTTCTACTTAGACGGGCACGCTGGCCCCCGAGAAAGAA
CATGCCGCGTCTAGTTTCACTATCGGTGCGATTTACCACGCGGACAAATGTGGACAGTAAAGCGGCGTGCGGG
GTTCTAAA"

,"AATCATCTAGAGCACTTTCTCAAATTCTAATGAGGCGTTCAAGACTGCTAGCATAACGGACTGTGTGTTCG
CTTCAGGTACAATGAAGACTTGTGGACACGACGATGCCATGATAGCGGTAGTGTTCTCTATGTCTCCTGTACG
CCCTACCCGGGATGTGGACCTCGTCCTCAGGGCGTGCGCCAGCGGATTAGACTCTTAGCCAATAATGTGTTTT
ATGCGCGCCTTGCCGCAAGAATCTCTTCGTACTGGATTAGGCGGCACGCCTGCGTCACGAGTAGAGTTCACAT
CGCGGCATCAATGTACTAGCAAACGGCTGGTGATACCTAACCGGCGAAAATCTGGAAGAGAAAAGCGCGCTGT
GCGGGGTCCAA"

,"AGTCACCTTTTAGGAGCTTCTCTGCAATATCTGGAATAACGGCCGTGGCGAGCCGCATGCTCAGCCATATC
GGGGACTGGATTGTGTCTGAAGAGGACAGAGAACGTCATTTTAGCAAGCGAGGCTTGATGATGGTGTTGTTCT
CTTGATCATGACCCCATATCTCGGTGCGTGGCCACTGGTCGTTACGTGCTTGATAGGTCTCAGAGGAGTAGAG
TCTCTACCTGAAATAGGCGCTTATGTTTCGCTAGCGACGTATCCTATCCTAGCTGTGACGCGCGCGTCCCACG
AGAAAGTTTCCTGCCGCGCTCATGATAACTCATCAGTTGGGTCAACCAGGCAGGAACAGTGGTGGAATCAGAT
GAGCGCTCTGGAGGGTTCAAG"};

    private static String s10390[] =

{"AGACTCGTGCGTTGGCATGACGCCGGTAGGGGTTTGATGTGACTACGAGCCGGGACACCACCAGGGGTTCA
CCCGAATCTAGTAAGTCGGAGCGCATAGATCCTAGGTTTGCGCGCTCGTAGGTGTTCATCGCCTCGCGAGCAG
CGGCAACCGGACTTCTGTGTACGCCAGAACGTTCCCGGAGCCAATAGGAAGTCTCAGCGGTAATCCATCGTTG
TAGACAAGGGGCAGATAAACAGAGGACCATGAGGGGTCGCAGAATAGGCCAAGTTCTGGGCTAACGAAGTAAA
AACCTCGTGCCAGGGCGGCAAACTCTGATACACCTGTCCATGAAATCTCCTCGGGTGGAGATTAGGGACCAAT
TCCGCAACTTCCAGGCTACCCGTCAGATCGTCAAACTTC"

,"ACTCACTATTATTAAGGATCGGTGAGACTGTTATGTACCAGACCGGGCGACACCGAAGGCGTTCCCCGAAT
TTCTTGTCTAGGGGCCATAACGATAGTATGGCGCTACGTGCAGTCTCATACTCGCCGAAGCGTGCGCGTAAGC
GGCAAATTGATTGTAGTCTCAATACCGTCTGCTAGGCGAAATGGAAGCTACGACATATGCTAGCATTCGAGCG
AAGGGGCACGGTAACAAGGACTCATTGTGGGGAGGTCGCGATGTAGCAGTAGCTGAGTAAAACGGTGATTACC
TCTGCTGGGAGGCCTAAACTGCCTTACGCCTTGTCATCGAAATCTCTCCCGGTGAAGCTAGGGACGTAATGCC
CATCTCCGGACTCCCCGAGTAACGTAACAAGTTC"

,"ACTATCGGGCTAGTAATAGCGCTTCTCGTGAGGATCGCTTGATGGGAAACGACCCGGCACCCGTAGGGTTC
CCCGAATCACATTATCCTAGTGCCATACTAGTTTTGGCGGCGGCTGGATCGCTCGTTCAGGGCCGAAGCTGCC
GCACAGATTGTTGCGTCATATACTTCCGGAGCCAATTAGAAGAGCTTAACACGTAAATGTACTCGTTACTCAA
GGGGCCAGTAACAACGTAATCGTGGGGGGTCCGAGCTAGATGCAGATTCGGGAACAGGATGATTCCGATCCAG
GAGCCAGACTCGCTAGGAGCTGCTTGCTAGAAATCGCCCGTGTGGAATGATAAGGGACGAACCCTCACATCGC
ACGCCTGCCGAGTACTGCCAAATGC"

,"ATATGCGCGGTGGTATAAGTCGCGAGACGCTGTGGTACGAAACAAGCGGGGCACCCGGAAGGTTCCTCCAC
GAAATCCTATTCTTTCGGGCAATGACATAGGTTGCGCTTCCATGACGGACTACTCATCCGAAACTCACCAACG
GCGTTATAGTCGTCTAAACGTTTCGCGGACCAATAGGAGAGGACTCAAGTAAAGTCGTTCAGTCGTGGCGGAG
GGGCGACGTAAACGGACACCTTGTGGGGTGCTGACCGATGTAGGCATGATCTGGGCTAAAGTAGTATCCTCGT
CCGGGGGCACAATCTGACTAGACCTCTTGAGTATACTCCGCGGTCAATGATAGGGGACGATCCGCACATCCCA
CGCATCGCAGTAGCGCTCAATC"

,"TAACACGGCTAAAGGCTCTCGGTAGACTGCTGTTGCATGCAGCCGGACCGAGAGGGTTCCCCCTCATACCT
TAAGTTCAGGGCCAAATACTAGATGACACTCGGATGTTCTAGACTCCGGATTGGCCGAAGGCACTTGTTATCG
GCTACGGTCCGGAGGCTATAGGGAAGCTTGGCATACACATGCTTCGTACATGGTAAGTGGTAAACAGAAACAT
GTTGGGTCGCAGAGTAAGGCACTACTGGGATAGCCGATGTCATTTCCGTACCGGTGGACGACAACCGCTTAAC

```
ACGCTTCCCATAGAATCTTCCCGGTGATATAGGGACGAACCACACATCTCACGGCTCCCCGAAGACCCGTTCT
ATA"};

    private static String s10400[] =

{"CGGGAGTAACCGGCAGACGCACAGCAGTGGCGAGAGGTGCAGGGTTTCCGGAGCTGAAATATGTTACCGGA
TACTAATCATGTTCGCTCTCTAGCAACTCGTTAATTCTGGATTAACAGCGTAACTAGGGTTTGTTGGTACACT
AATTCTATCGAACGTAAACGCTGATCGCTGGATAGACGATCGGATACGAGTAACTGTCACTTCGAGGGCCTAA
CGGACAAACTTGCATCGTCGACGATGGTCAGTTCACTTGTGAGACAACTTCCAATGGGTGCGGTCATACGCAT
CAGACACCGCTAAGCCCGTTCTCGCGCGTACGTGGCCGGAGAACAAAGCGTACTGAGCGAAGCTGGGATCCCA
ATAGTACCGACATGCGTTAGCTTATGTAGCCTGAG"

,"CGAGATAATCGAAGAAACCGACTCAGATGGTCGAAGGTTCGAATTTCGGGGTTCAGAAGGTGGTTACGCTA
GAATCAAGGTCGTTAGCGTCCGCAACTCGTTAATATCGGAATAACGGCTGACGATGTTGTCTGTTTGAGCAAA
AGTCAATACCCTTGAGAGTTAAAGGCCCTGTGCCCTAACACGAGATCGGTACAACGTACTGTAATCCACGTCG
TTGACAATTCAGAGTAGATGCCATTGTTGTTACTATGTGAGAGCACTGATCCAATTGGGGCGTGCGACTCACG
CATCAATACGACCGCTAACCTGTAGCTCGCGTGTATCAGGACTGAGAACCAGAAGGTTACAGAGTGCAATGCG
GTCACAAATCGTTCGCGGACTACGATTACCTATTTGTTCAG"

,"GCGGAGTAAACGGAAGGAACTCCCCGGGTTGGCAGAAGCTCAGATTTGGCTGATGAGTATAGGATGTAATC
CGCATTTACTATTTTCGGTATCTCAGCAAACTCCGTAGTTCTGGATGTAACCGGAATGGATTATTGTTTATAT
CAAAGTCTCTACACTGAATGACAAAAGCTCGTCCTAGTATACAGCAATGCGAACTAGCCTAAACGTAGCTCGG
GGCTCAGCATCAATTTAGATGGACCGCCGATGTTGTTCCATTGTGACACAAAATCACTGGGCCTTACTACCTG
GAATCGCGCGCGCTAAATGTACTCGCTACTTGGACCGTGAATCACGAAGATTACAAGGTGCAGTCTGGGTCAA
TCCGTATCCGACATCGTAATGCTTTTCGTTCCTCTA"

,"GCGGACTTAAGCGAAGAACGACTCAGGTTACAAGGGTACTATAAGGATGTCGAAATGGTGTGTACATAGTT
AATTAGCTTTGCCTACCCGATAACTGCTGAGATTTCGGATGCTAGACGGCATCATCGTTTGTTCTTTAGTCAA
GATTCAATTCCATATGTGATAAGCTCTCCGGCTATGAAGAAGTCGGGACAGTACATGCACATCAGGGGCTCCA
GCAATGTTGACATGTACACGGCGATTCTCCTTATCTGGTGAGCACAAAGTACCATAGGCGCGAGAGCCCGCGA
CAAGACGGCCGAACCGATCTCCTTGGCACGTTGGACCGGTAACCAGATTGTATACAGACGATGCGTGGTCCAG
AACGTACAGTCGACACTCAGTGCTGACCTTTTGGTAGA"

,"GCGAAGTGAAGCAAAAAGCCCCCGGAGTATCAGAGGGTTCAGGGCTTCGGGGTTGAAATGGTGAGTACCCT
GAGATACTATTTTTCGCTACTTCAAGCAACTGTGAATTTTCGTGATACGCCGTACCATTTTTTGGTTTAGCTA
AAAGGTATTCTATCTAAAGTGTAAGCCATGTCTCGTGATAGTATGATACGGATCACAGTACATGATCTCGGGG
GGCTCAGCATAATATCGCTGTATGGAAGGCCGAATGTTTGATCTACTGTGAGCACAAAGCATCCGGGTCCGGT
ACTACCGCACATTCCGCGCCTAACTCGCATCCTCTGCGTACGGTGGGATCGATAACCCGCTGTCAGAGTCGAA
TGCTTGGGTCGCACTTAGCCCGCTACTATCACATTTTGCTCCCGA"};

    private static String s10410[] =

{"GCTTGATCGCTACCGGCAATGGACATGAAGAGAGAGATGGTGGGCTATGCTCCATTATGATGCCATATTGG
CGGGCCACAACTAGATCACAATGTTGTCCGTAGGACGTTCTAGTATTGGCTTTACCGACGTACGGCTAAGCAC
TGAGCCGCCCTGGGAAGGTACCTTCTAAGGTTCGGAACCTTACTCTGATATTCTGGCTGGGCGCGCAGCCCAT
ATCGTACCGCTGCCTCCGCTGGGCATGGAATAAGTACTTGGGCGTGCTCTCACGAGGATAGATTACATTGTAC
TCGACCGATAGACGGCGCACAGAGGAAACTCAGGTAGGATGCCAAGTTGTGCGGCCAGGGCCACTGCTCCTGC
TGCGAAGCTATGTGCCAATTTGTTCGCATTCCGCGATTTCATCTCTCGGT"

,"CGCGGTCTGCACGTATAGGAACTGGCGAATCGTAGGAGGGTCGCTAATCGCCTATTTTGTGGCACCACTGC
GGCGTTATCAGTTTTACACGGGTCAAGTCGAACGTCAGTACGGGCCTCCAGGGCGCAATAGACTGAGCGCCGC
CAAGGGCGAGTATCGCAGGGTCGTTACCTACTCTTTGTTATTCTCGGGAGGGCGAGGACCCATATGCCTAGAC
CCCGCGCCGCCTCGCGTGGACCTGGCATGACTCTGTAGGCTCGCTCTAGCCAGAGGGGAGTAGCTTTTAGCTC
CGCTGAGCGGTGACTAAAAGGCAACAACAGAAGCCGTAGTCCAAGGTGTGCGCCACACGCGCCAGTTCGCCTG
TGGAAGGATAAAGTGCCAATTTATTGCCTGCAGGGCTGCTCTACCGCACCGTG"
```

```
,"GAGGGGTCTCGCGCTACTAGGACAGTGTAAGTTGAGAGGTGTGGGTGCTCATTTCCGCGCACTATTGATCG
CCAACTTGTATCAACAAGGTTGACTTGAACTTCTTAAGCAGCCGCGCGGCGTCGAGTTTACGTAGCGGCGCCA
CAGGGCTGGATACCTGCAAGAGCTATGACCCCTACTTGTAATTCCGTGCTGGGGGCAGCAAGCGCACTTAGCC
TAGACCCGGCGCCCCCCGGTGGACTGAGATACATTGGCGGTGACATCTAGTCGCACGGGACGAATTACCTTGA
CTCAGACATCGAGAGCCGTACAAGGCAGTAACATGTGGCCGAGTCAGCGTGCCGCAAGGCGCAGGTTACGCGT
CGTACGAACGTAACTGTTACATTTATGCTGACGACTGCCCCGTGTACGCTCCAGTT"

,"GCGGCGCTTGCACGTATCTATGACATTGGACAGTTAGCGAGTGTGGAGCTGCCTTTATTAGTTGCATATCT
GGCCGCCACATAGTTCCACAGTTGTCTAACTGAACTGCTGTTGGTCTCCGGGACAGGTTCGAACTGAGCGCCG
AACAGAGCAGTGAGATCTGGCATAGGTACATAAGCCATCCGTGATATTATCGTGGGGACGGCAGGCCGATTAG
TGCATACGCCGCGCTCCAGCGTGGCACTCGGGATAAGACTTGCGGTGCGCTCTACAGAGGACTAAGCTATTGC
ATAGAGCATTAGGCGAGCGTCATAAAGGATACCCGAGGCGTAGCGCTGTCGGTCGTCCAGGCGCGATGTGCTG
ACGTCCTCGATCATAGTGCCAAGATTTGCCTGACTATGACTGCTACCGACTCTATG"

,"GCGCCGGTCTGCCTGATTAGGACTAGATGTGGGGGTGGTGGCTGTGGCTCTCATTTTGTCGCATATGTCGG
CAGACACTGTTTGAACATGTGGTCAGCTGAACGCTTTGAGGGTCACCGGACCGGTAAGTCGTCCGCCGCACGA
GCAAGTCCTGAAGAGTGCCTAAATCTAGCTTGTATTGGTGGTGGCCGGAGCCTTATGCCCTGTCCTGCGCGAC
TCGGCGTGGCCAGGAATAAATTTGGGGCTGCCTTCACCGTAGATATCCAGTCATCTCGATAGTCATGCTAACA
AGGGCAACCAGATGGCGATGCCAACGCAGTGGCCTAAAGCGCATTGTTCGACCGTCTTGGCAACGTAACTGTG
ACTTATTGCTCGCTATGCCTCAGTTAAGGCATCGAT"};

    private static String s10420[] =

{"TAAATGTAGCTCTTTGCATAGGGGGTTTTGAGCAGCGTTACTCCGCGGTTTCCTCATTCGAAGGCTGTAAC
GTGACTGAGCCTCGAGTCTTCGTTAGGCGTGACGAGTAGCTCTCAGCGGTCCTGCCTAGCACTTACCGAGAAG
GGCCAGTGCGCCATTGTAAAGACTCTTCCATGGCACAGTGCTCCGTTCGAAACGCCCACTGTTTTCTTTATTC
TACGATTTATTCGTGTATCGCACAAATGGCAAAGGCGCTCCGGGAACAGAGGCACCACGCCCGGGTAAAGATT
AAGCTAATGCCAGCAAACTAGGTAGATGTCCACGCTGTGTCTGCCAGACGCTTTCGTGGCTCCTCGAGAATGT
ACTGTGCATACTAATTTACAGTCGATAACGCTCAAGGCATCTTATCTTTGGGGC"

,"ATATTCAGGGGGCCTTGGCCTTGGTGTTGTTTAACTCTCATTCTTGGTAGCTCCATCTAGGCAATGCATTA
ATCGGGCGTCCGCCCTTAGCTACGTCTGATACGTAGGTGTACGTCAGGCGGGCCTGTTCGACCCTCCTACCAC
ATGAGAGGCACATGGCGCCGTAGCACAACGAAGGCTACCATCGTCAAACCTCTACCTCGTAAGAATCCCGTTG
TGGTTGATGACCACCAGTTTGTCGTGAATCGCGATAACGGCAAGCGCGATGAGTGGAAACGATCGACCACGCC
CTAGGAAAGTTAAGAATCAGACCGTAATCTGCGGGTATTCCACGTGGTGTCATACGGAGACTCATTGTGGTCT
CAGACTTGACATTGCGTAAAACATGTAATTCGTATCCTCCACGGCAGTCTCTTTAGG"

,"ATATTAGGGCGTTGGGCCGTTGCGGTATTTAGACTTCATCCCTGGCTTCCGCATCTGGAATGACTTGACGT
GGCTCGACGGCACGAATTCGTGTTAGCCGTGAGCCGATCTTTCAGGCTTGCGGATCAGCCACTATCACGCGAT
GAGAGGCAGGCGCCCTATGCAAACAGGCTAACCCATTTGACAAGCTTCCGCCTGTTCGAAACTGCTTTTGTGC
GTTCCTACCCATGTATATTCGTTAGTCTACAAACTAGAACGCCGCAGCGCGGAAACAAGACCGAACCCAGGCC
CTGCAAAGATGTAATGAATTCAAACGGGAAAATCAAGGTGGATCACGGTTAGGTCCGCCTCCCTTATTGATGC
TCCCGAACAGCACGTCTGATTGCTAAGTACGTCGAATACCGTACTGGCGTCGTCATTTAGG"

,"TAATAGAGTGGATCTGGACTGTGCGGGTTTTACTATCGCACTGCTGCTTTCGCCATTCGCGAAGGTGCTGT
CACTGCCGCCGGCCTGCGAATCCCGTAGCCTGGAGCATATATTACGTGCCCTGTTTCGACCATCCTACCAGTT
ACGGGCCAGGCGCTATGGAACGCACCGTCAACCATCTGGCATACGTGCTAGTTGAAACCAGCTCCGTTGCGTT
ACACCCGATTTTCTGATTTACTAACTTGGACGTGTGCCGGGGGGAAACAGAGAGCACCCAGCTCCTGAGATAT
AAAGATAACGACGGAAACGTAAGTGGATGCGCAGTTGTGTCGTCCGAGAATCTGTTGCTCCCGAACAGTGTAG
TTCAGTAATCATCTACGATCAATACCCACTATGGCAACTTGCTTTAGGG"

,"TCATTAGGGGACACGGCACTGGCGTGTTTACACATCCTAACTGCCGTGTTCTCCATTCGCGAGTAGTCTAG
AACTGGGCGACGGCCCGAGGTGTCGCTTAGCCGGTGACGCTGTCGTTCAGGCTCTCTTCGTCCAGGCCTCTAC
CGATCATGGGCAGGCGCCTATGAAAGAGTCTCACCATTGCAAACAGTTCTCTTTGACGCTCACGTTCTCAACC
CATGTTGTTGCATGTTCATAAACAGTGGAAAACGGACTGCCGGGAAAAGTAGACCACACACGCACCTGGAAAG
```

```
AATTAAAATGCCACACATAAGGCTAGTTCCAGGTGGTCAGTCGACAGCTTAATGTAGCTCCTCCGGACATAGC
TGTGCGAATCTATGTCACCGGTCTATAATCCTACTGGCATGTCATTTAGGT"};

    private static String s10430[] =

{"GGAACGAGATCCTCTCCCCTCCTTGAGAGGCTCCGTGAGATTGTTAAATAATGTTGGGCTACTCGCTTATA
TGTAGCGGTCTGAACCCAAAGCAGTTGTATTTCCGGTTTGCTAATGCCCAGAGTTTGGACTTAGTTACACTAT
ACACAATATCTTAAGACTAGGAGGCCGCTAATCTTAATGAGGGGTGGCAGACCAAATGTCGAGTATAGCACTT
CCGCAGAAAGGGTATCGTTCCCTCGGGCGGGATAAAGTAACGATTACCAAGAGGGGTGGAATCGCACAACTGC
CGGCCCACGGGGAGCTATTGAATCGTTACTTGTATAGCTGCCCGTGGGCTCCAGACATAACCGTCGAACGGAA
CTATGATTATTGTCGTCTAAGTTCTAGCTCGATACGACGAGGGACATTGTCGTTTTTGATTTCATCCTTT"

,"GCAACGAGAGCTGTCGCTCTTGAAAGGCTCGGGAATTTGTAATTTAGTTAAGACTGGCTCGCATCGTTCCG
GGACTAAACCCTACAGCTATGGTCCTATTCCCGGTCGAACCCCGAAGGATTGCCGGTATTGACCACATATAAC
AATAAGTCTATCCGAGGGGCACGCATGATCTTAGTTCAGTGGATGCGAATCCAGTGAGTCAGTGTTGATTACT
GCCGAAGAAGGATCCGATCCGTATAACGAGAAAGAAGGCCGTAGGAGGGTACTTGGGACAAAAGACTGCCGGC
CACCCGCAGCTAATGTTGGCTGCATTATCTTCGCGTGCGGGCCCACGAACTCAACCGGCTAGAGAGCGAGTTA
ATGTACTACTTTCTAGGCTATGCCGTGTACGAGAGTGGGAGGCATGTTGTCTGGATTTATCTT"

,"GCCTGGAAGGCTGGTCGTAGAAGAAGGTCCGGGGAATTGTTCACAGTTTATTAAGACGAACTTCCGGCTAG
TATTTCTGCGAGCTGAACACCCAGCATGATTATTTCCGATGGTAACCGCGAGAAATTGCCATTATGACCATCA
TCAAAAAAAAATTTAACATATGGGCCGGCGTGTATTCTTATGAGTGCAAAGTTGTGCAGTTGGCATCCTGCCGA
AGACAAGATCTGTTTCTGTACCTGAATAAAAACGCGTACCGACGAGGTGATTGGAAGCCGAATCGCCGGAACC
CCGAGGCAATATTGCTTATGCGTATCATGCCGGGGTCGAGCTATTAAACTTAAAGACGTAGTGAATCGTTTCT
CAAGGTTGCGCCTCGTAATGACAGTGGGTGCAAGTTCGTTTGAGTTGCTT"

,"GCAATCGAGACGCTGCCGTCCTGAAAGGCTCGGTAGATTTGTAAATTATGTAAGCGGCATCCCGTACGGTT
GCGCGATGAACCACACGCATGTATTTTGATGCAACGCCTGAATGATTGCAATACGTCACATAAAAACAAGATC
ATACCAGGAGCCGCGGATTGCTATGTTAAGGGGTTGAAGATGAGAGGCATGTGACATACTAACGCGAGAAACG
GATGCTGGCCGTCCGGCGGGATAAGCGCCGTTACCGGAGGGTTAGTTGGAACTCGACATCCCGCCACCCTGAA
TATGATCTTCTGTATTAATGCGCGGCGGGGGCCGATTTCTAGCACTGGCAGAGGGTAAGGTATGACTCATTGC
GGATTTAGACTCGGAGCGCGGTGGATGAGCTGTCGTTTGATTAATCTTT"

,"GCAAGCGTGTGACTTTCCTGCCCTAGGACAAAGGCGCGGTGAATATTATGATGTAAAAGCTAGCTCCGCGT
TGCGTTGTGCGGGCGTAGAACCCAAGGCATGTTATTTTCCGATCGACCCCGCAAGGACTGGCATAGGGTCAAC
ATAAAAAGCGAATATTTAACTGAGGCGAGGCTAGTCCTTAGTGAGGATGGCGACCCAAGTAGTCGAATGTCAA
TTCTGCCGAGAGCACAGGGAGGCCTTTTCTCGCTGCCGAGTTAAAAGACGCCGATCCGATGATGGGGAGTTGA
GACACATGCATTGCCGGGCGACGCGAGGCTAATGTCTTTCGTTATTACTGCCCTGATCGGGTCGCAGAGCACA
CTGGAAGAGGATGTGATCATACGCGTTCCTCAGAGATTATGCCTCTGATCAGAGGGGATGCTACGTGTTTTCA
TGATCTATT"};

    private static String s10440[] =

{"TCTCAGGCGACGACCAAAACATAGTCATCAAGGAACTTTGACGACGCCTGGTGAGTAAGATTTTCTGTAAC
TCATTCCGTGTACGAGGGTCTAAGCAACGTGTTTCGTAAGAAAAGTCGCACGCTGGAGTTGCCGAAATAAGGA
GTGAGCCAACCGCTTCTGATAGAGGTGAGTGTGAATGAACGAGCAATCATTGTAAGATACTTTGCCCGTCGAT
GCAGCTTGCCTGATCCCGTGGAGCTCTTAAGTGACCTGATTCTTGCCATCTCGCCAAAATAAGCACCCGAGAC
AGTGAGAACGGCGGGGCCTAGTTTAAGTAGCCGCCGACTAGCAGTGGGTGCTACTCCCGTCCACTATTCAAGG
AAGGCGGAGACTATACGATTGCGGACACCGCTATTATCCCATCCCAGTTATATGAATCGTCCGATGCGGCACA
GGTACTC"

,"CTTAGGCCACCCAAAAGAAGGGAAAATATAATCTGCCGTACGACGCAGTGCAGTAGACTTTTTGTAACCTA
CTTTCCTGGTAGAGGTGCTACGAGGCGTCTGTAAAGAAAGACTGCATTGTGTATGACTAGATAAAGGGAGCGA
CAACGCCTATCTTAAAGAGGGTGTAGTGATGTATGCTATCTATTGTGAAGGATTCACTGTCGCGCGCGATGAT
CGCAACCTGACCTGGGTAACGGTTATCTGTAAGTGAGCTCTGTCACTCTCACAATCATGGACACGGCAAGAAG
ACCAGAGGGTGCCCTTGTTAGTGCAGGCGACTGCCCAAGTAGGGTCGTACCCTCGCGTCCGTCTGTGTCAAGG
```

```
GAAGACAGGAACATACGATGTCAGAACTCAGGTTCCTGTAACTCTGTACTGCGACGTCGTCCAGTGCAACGGC
ACTC"

,"TCGATGCCCACAAACAGCCATCTGAAACTTGCATGACAGCCTGGCAAGTAATGATTCTTCTCTCAACCGTA
CTCCGCTTGATGCGAGGTTACTGAAGGTTTTGTAAGAAACGCCAGCCTGATGACATCCGAACACAGAGGCCGA
AAACCTATCGAAAGGGGTAGGTGAAGAGGTCTACAATTTTAGAAATCATTGTCCGTCCGACTGCGTTCGTCCA
GACTCCAGTAGGCCTTAATTGAATGCGTCACCTATGAACATCCACCAAACTGAGGGTCAACCGCAACGGCAGA
CAGCGGGAGCCTGCTGTCACGCGCGAACTCGCCAATTGGCGGTACGTCAGCTTCCTGTACGCATGATAAGGAA
GGCAGGAATTAACGATGAAGACCGTTAACTCCCTTTAATCGCAGGCTGGTCTGATCTGAAACGTCAC"

,"TCACACGCCCGACCAAGAAAGACCACTCAGCAACTTACATGACAGCGGGTGAAGAGATTTTCGTTCAACTC
ATTCCGTAAGCTGATCGACGTGGTCGTTATGACAGCCGAACCTGAATGTACCTGGCTAAGAGGGGCCACTCCC
CTCTTCTAAAAGGGGTGAGGGTGCATGAACGACTTACTTATTTGCAAATGGGATTTGCACTGTCGTAGGCTTC
CTTTCCTCTGACTCGAATGCCTGCTTTTCTGTGAGCTGATCTCTCGGACACCTCTAAAACATAGGGAACCGGG
ACAGGAGAGAAACCGGGGAGCCGTCCTTGTTGTCGCCGGGCGATCCGCCATGAGGGTGGTACTCCCGTTCACA
CTAGTTCAAGGAGTCAGGCATATACGATTGACAGGCACCGTTTATCTCTAACCCTTTATAGCAGCGTCGATCC
AGTCCCGAAGCCATCC"

,"TCTAACCCCCGAAAACGACCCATAGGAACTTTGAACTAGACGTGCGTAAAAAGGGATTTTCTATAACCTAT
TCCTAGATTCGAGGGTGCGACCTGGCTTAAAGAAGCGGAGTCGTATTTGTCCAGAAAAGAGGCAAACCCTCGA
TCTAAAAAGGGTGGAAGGTTAGATACGACTATATTTTGAAATTCCATTTGCCGTCGGATGATCGTCCTCGCTC
GCATCGTGAGGCTTCATGGTCGATAAGCAGATCTGCAACCTTCAGAAACATAGGAAGACGAGACAACGCGGGG
GTCCTCTATTGATGCGACCAGACGCCGCCAATGGGAAGTCGCACGCCGATCCACGTACGAGGAACGGCAGAGC
CTACAATAGCAAACACGCTTGTATTCCTATCTCCTTAAAGCACGTAGTCCGATACGGAACACCACTC"};

    private static String s10450[] =

{"AGTAGTTCGAACCTTTATAGCAACTGCCATGCGCATGCTTCTAATTGTCAGACTCCAGGGGGTGGCGATGA
TATCGTTGAATTGTGCAACGCAGACGATGGACGTGTGATGTAATCTTTAATATGTACGACACGAAAAATCTGG
GATTGTACGGGAACGATTTATGAGGACCACTAACCATTATACACAACCTTAGTATAAGGCTAATCAATTTGTT
TAGTAAATGTATAAATCTTTCGATGTTGAGACCTTCCTTGGCGACGTAGTTCCGAGCGCTTGCTCGTCCAAGA
TTACACCACATCGTGTAGACTGTTGGAGGGTGCTCTGTCCTCCGTCGGTAGGACGGCGGCCTCAGTAATATCT
CGCCTGCAGTAATTCCTACAAGTGAAACTGGACGTCCCAGGCAAGTGTATATTTCTGGCACAAGCCCATGGAA
A"

,"ATGCAGCCACTCTTTGACGCCTTAATGCACTGCAACGGTTAGCCGCTTAATATTTTATACCCAAGGGCGAG
GCGCAAACAGGTTCATTAGAACTGGTCACGAGAGCGATGAATGAAGGATGTTAATTCATAAATGTAAGGACAA
ATCATTACGTTAGCTTTGCCAGGGGACTGTATAGGCAACCATATTATAGCACAACACCGTAGTGTAAGGGGAA
TACAATTTATTTTTTAGGAATGAACCTAAACTTTCTTGTATTCCTCTTCTTCCGGGATCGGTTTCAACCGCTG
TCTGTCCCACAATTAAAACAATGTATAGTATGATAGGTGTACCTCAAGCTCCCGACTCATTGGACAGGGCGCC
GCATTGAAACTTCACAGGCTAAATCTCTAGGACGTTGAAAACTGGAGTGCACGAGAAGGTTATTATTGGACCA
GACCCAATATTGGAACA"

,"AGAGGAACCTTTCGACCTTGAAAGTCATACATTCCGACGCTTAGGGGTACTAAGTGTCTATCAACCGGGGT
AGGCGTAACATTAGTGTAATGTTGCACGAGTCGAGGAAGAGAGGATCGAATCTATAAATCCGAACTCTTAAAA
TGCTGATTTGTCAGGGGAACGTTATATGACGGACCATTCAACTTTCAACACGCTTAGTGTGAGGGGACGTCAA
ATGTTTTGAGGATTAAATAAAGTCTATTCATTAGTCCTCTCACGCTGAACTGGTTCCACGGCCTATCGCTCCA
GCAAGTTAACACTACATTTGCTGATAGGTAGTGGATCCTTCTTGCCTCCTTCGTAAGGAGTGGAGGCCGCCTT
GGACTACTCTGACTAGATTCCTAGAGCGTTGAACCGGACGGTCGCAGCGAAAGGTGATATTCTGGACCAGCCA
TCAATAGATAA"

,"AGACAAGGTCTAGACCGTTCTGGTACTAGCCACGCGCAAGCTGTCTTAAGGTGTTCATACTCCGGTGGGTG
GCGAAAGCTTTCGTTGAATACGTTGCCAGACTACGGTGGATAAGTGGATATATCTAAAATTGAGAACACCAAA
AATCGTCAATTTTAGGGACAGTTCCATCGGACAACCATTATCAAACACCATTGATGTAAGGAGCCATACAATC
TATTTATGAAAGCAATAAATTCTATTGATTCTCTCTTGCGGTCGGGCCACGACGCTCTGCTCCCCAACGATTA
AACACATATTTGGATACTTTGAGGGGTACTCCTGATGACCCTCCGTTGTAACAGAACGTCGTCCTCAGTAAAC
```

```
TCTCTTGACTGATTCTCAGGAGCTTGAACTGAGCTGTCCGAGAATGGATATTTTACGCACCAACGCCATACTA
TGGATA"

,"AGTACACGACTGCTGACGCTTGATACCACGACCAGCAGCTGATTGCATAGTGGAATTCACCAGGCGAGTCC
AAAAAGATCATCTTGAATCATGGTGCAGAGCCAGATGCGAGAGATGGTAGCACACTTATCAATGGGAGGATCA
CTCAAATCGTCGTATTTGTCGGAGGGACTGTGTTAAGGGCGACCAATCGAATATTCAAACACGTTTAGTTGAA
GGTAACAATTATTTTTAGAAGTAGAAATAATCTTACCATTGATTCCTCTGGTGAGCTGGTTCCGCGGTACTTT
CCTCGATATAAACTCGAATTTGGAGATATGAGGGAGTACTCTGTCCGCCCTCGTGTTCAGGGAACGGCGCTCA
TGAAACTTCACTTGACTGAAATTCGATGGAGTTGAAACAGGACGTGGCCCGAAGAGAGTATATTTTTTGAGGC
ATGCCCAATCAGTCATAA"};

    private static String s10460[] =

{"AGCCTAGGTCACACACTCGTTCGAATAGCCTCATCACGACACAGAGGAACTACACTTTCTATTATAGCGCC
CCTAATTAGTAGTGGCCTGACGAACCGTGCAAAGCTGTACACACATGGCCATTAATTCATTCTTGCCTTGGTT
ACTGCGCCCAACCTGCTTGTCCGTAAGCAGCGACGGCAAGTTTACCTGGATAACTCTCCGTCGACACATTTAA
ACGTATCTTTTGGAGGCACCAGGCATGCGCGTCTGAGGTATTACCGTAGAAGAAATGCGTATGCGATCAGAGT
AACATAGTATCGCTCCCGTACGTTTGAAGCTCCATCCATCGCCCCTACTGCGTTGTATGGCTGAGAAACCCCA
TCGACTTGGAACACGTTCGCTGCACAGGCGTACACCCAACCGCAATAAGCTGCCCTTAACTACCGCGGGAGCG
CGCCTGCAAGTAGAGCGGTGGCCT"

,"AGCCTAAGTCACTATCAGTTTAAATTCCTTAACTGCGGAACCCGGAGCATAACATTTTAATCATAATGGCC
CATATGGATTGTTGCGCTTACGACCGGTTCAATGTCGTTAACAGCACTGCTATAACTTATTATGATTGTTTAG
GTGCCCAACGCCTGTTTCGCTAAGACAGCGGACGAGCAAGTTTAACTACGAAGCTCTTTCACCAATTAACCTT
ACTCTGCTAAGGGCACCGGCTATCGGTTGCGGGGGATTTTACGCTGAAAAAACAGCCTCATCGGAGGGGGGAC
ACAAGGTATCCGTTCGCCTGATAGCCCTTGGATCGCCACTACACGCCATCCAAATGTTTTCTGACGGAGAAAC
CCCACTGTACTAGACGCTTTCTTGAATGGCTAGACCAACGAAAAAGCCTCCGCAAGTAATCCCCGTGATATGG
GCCGTACCATATGTAATGTTACGCGCCGGGAT"

,"AGCCGAGGCACGAACTAGTTTGAATACCTTATCGGCCAACGCTCGAACAATGCCCATTTAATACGTTAAGG
CCCTTCATATGAATTGGCCTGACAATCATTCAAATGTGAACCAGCCAGCGCCTAATTCATTTGATTGCGTTCG
ACGCCCAAGCGGCTTTGCTAAAGCTAGCGACGAGCAAGTTTTCATAACTCGCTATTCCACAACTTAAGTCACC
TTTTAGGTGCACCTCGATGGGTTGCGCGGTTTTTTTCCGAGGTTAAAAAAGCTGAACAGCGAACTACGAAGTT
ACGGCTTCTCGACGCCCTTGGATTGCATCCAGCCCTACGCGGTTGTCGACGAACACTCACACTACCAACTAGT
ACCCTGACATGGCAGAAACCCACAACCAAATTAAGCTCGACATGACTACACGTGTGCGGCCTCACACTAGTAT
AGTGGCCTCCCGGGACC"

,"AGCTAGGCCCACGCAACTGTTGAATAGTTATTCGACAGCACTGGGACATACGCCAATTAAACTCTCAGCCT
CATTTATGTAGTTGGCCTGAACCACCCATTCAATGCTGTACACACAGCCGATATCAATCTCTTCTAGATTGGT
TACGGGCCCCAACCCTATTGCTCGCTAAAGGCGAGCTGACAGCGAAGTTTATTATCAGCCCTTCCCACTTTAA
GATACTTTTATATGCCACCGCGACGCGACCCGTCGGGGATTTTTACGGGTAAAACAATAGCAAATCGACGAGG
CAACCAAGTATCGTTGCACGGCGCCCTTGTTTCGCACTCAACCCCCCACGCGCTTGAGTGATCGGAACACCCC
TACCTAAACTGTCTGCAAGCCTGACCCCACCAAATTTAGTCGCCGATAACACTGTATACAGGACCATCCATAT
GGGCGTTGCGCGCGGGCT"

,"GGCCGATGCCCGACAATGTTCGTATATACGTCTTCTGCTACACCCATGGGACTATACGACCATTATAATCG
AATAGGCACCTCTGGTTTATGCTCCTCGTACACCGTTCGAGTGCGTTAACCCGCGATTAATCTACTTGCTGTG
TTTGCTCGCCCACGTTCTTAGCTCTAAGTCGAGCGACCGGACAATTTACCTGAAGTACGTCCGTCTCACAATA
AGTCCACTTCTAGGACACCGCGACTTGCGCCGTTCGGGGGTTTTATCCTGAAAAGCGCATATTCGAGAGGCGT
GACCAATTCCTGCCAACGCCTTGGATCGCACCAGCCCTACGCTATTTGATACACAGAATAGCTCGCTGATCGG
AACCATGCTCGCAGACCTGACATCCCATCCAGAATTATGTCGTAATCTCCCGTTGACGGGGGCCGTCACATAG
TGAGGTGCCGCCCGGTCT"};

    private static String s10470[] =

{"CATGAGTCACCATGAGTTTGGCTCGCACAGTAGCTACCTGTCTTCTACAATTAACACTATCCCCAAGATGT
```

```
AGACTCCGGTGATCAGCTTCGGGAACGCACCGCTATCGAGGCTACCGCTCCCTGCGAAAACTGGAGCGGTCTC
AATACCATCAGCGATAAAGCACGATGCAGAGATTTACCAACAGCAGCGTATGATACTCTCGAAGTAATTCGAG
TTTCTGCGACGGTAACAGACAGGTTGCGTGACTCGGGGTTGAGTTAGGGGGAAGCTGCCTCTCTTTTACAGCA
TAACGGTAACACCTAGAGTTACGGAAAGCTCACGATATGAGGATTTTGTCGCAAGGACGTACTCGACACTAGC
TGCAATAATGTTCAGTTGTGGCGTACTCGGAGGCACAACCCGATAGAACCGAGTTTCGAGCTTGAATTGGGCG
ACGACGATAATGAACAATCCATCGAACTCCAGTGGT"

,"GTGAGCCACTATGAGCTTTGGCCCGCCTGATGGATCTACCTTTCTCTCCTATCACAATACATGTCGCGAAG
CTACTACCAGGTACACCCCGGACCGAACGCCGCCGAAGCGAGAGTCACACCTGCCCTGGTCAGAACAGTCCAC
CGTATCCTATAACCTGCTGATTACAAAGCCACCTAGTCAGGATAGCCAAGACGACTGCTAAGATTGCGATAAT
AACAGTGTTTCTGACCAGTAAGCAACGAAATCTCCGTAGGTCGGGTGTATGTATGGGTGAGCGTAGTTCTTTC
ACAGCATTGGATGACTGACTTGGTACGGAAGACTCGGTTATAGCTTCCTTCCACAGCCGCTTCAAAGATTGCT
GCTAGATAACTCATGGCGAGCAGATTCAGGAGGCACCCCAAAAAAACTTTATTGCAGGCTCTGAATTGAGGCC
GACGAAAATCATTCATCACAGATCACTGGGAT"

,"GATCAGGTGCACCTCAGACCTTTAGCCAACACAGGGACCACTTTTCGCAACTCTCCAAAATCGTTCCCGCA
AGTGCCTTCCAGTATTCGCGTGCTGACCAACGCGGCAGGCAGGGCACCATTGCCACGGTAAACATCACAATCC
CTAATCAACTACGTATGAGAGCAATGACGCGAGGATATCCCAGCGCTACGCATAGCTCTCAAGTATACTGGTT
CAGCACTGAAGAACGAATTCCGTGATTTCGGTGGTTGGTAGGGGAGAGCGGCGATCTCTTCAGTATATTGCTT
GACTACGTAGTTACGAGAGATTCAGCTTCAGTGATCTGTTCACCAGCACGCCTCACAGCGTACGTGAAAAAGG
CATCATTGGGAGACGGACGTTCGTAGACAAACCACGTAAAGAGACCGTGATTTAGAAGCTTGTAGACTGGGCT
GCAGAAATTACTGATTCCCTCGAGCACACTGGGT"

,"AAGCAGCATTCATGAGCGTGTCGCTGTCAGCGAACGGCCAACTTTCTCTCCACTACCTAGTTTCCGCAAGA
ATGATACTCCACGATCGCTCGTGACCGACGCGCCTGAGCGAGCTAACCTGGTCCTCGCGAAACCTTAAGCCGA
ACTGATCACTGACATGATATAGTCAGGTAGCAGGATAGCCAAGACGGCCATGCATGGCCTTCGATAGATCCCG
TAGCTTCAGCACAGTAACCGAACGAATCTGATCTCTGGTGTGGTGTAGGCGGGAGAGGTCGACTTCTTCAGCT
ACTGGATAGTCACCGAGTTCGAGACATTTAGCCTTTAGTGGCACTATGTCTCAAGCCAGTGCCTAAACAATGC
TGCATAAAAGCATCGTTTGAGCCTCACTAACAACACAGTACCAAATCCTATTGTCACTATGAATGGCGGCGCG
AAATATCAGACATTCACCAGAAACGACTGGTG"

,"AGTAGCAGCTCGAACTTCGCCCCAGCGGAAGTACTTTTCCTCATCTCACATATCCCCTGAAGTTGCAATCT
CCGGACTAGCCTGCTACCCAGCCGTCGAAGCAGGCGACACTGGACCCGTTAAGACTACTGATGGCAATCTAGA
CACTAGCTGTAAAAGGCCCGAGTGACGAAGATACACAGCCGCATGCTAGCCAGATCCATCGGACATCATACGA
TAAGCAAACGATATCGCGATGTTCGTTGTTGTGTTGCGGGCAGCGCTAGCTTTTCAACATAAGAAGCATACAG
TAAGTTCATGGAAGACTTTAGTCTTATAGGACTGCTGTTCTCAGCAAAGCCTCAATCTGGGGTCGCAATAATA
TGTAACGTTTGTGGAGTAGCACTTGGAACGACACCAGTAGAAAAACCCAGTTATTGCATTCTGAAGAGTCCAG
GCAAAACTGAATCACTCCAGAATCACCGGTG"};

    private static String s10480[] =

{"CCTTCGGCCCCACACAGGACTGTGTTTCATAATAATTCCTCCCTAGGAGTGCACCCGAAGAACACAAGGTT
AAGGCGTGGTCGTCGGTCAAAAGTCATGGTTTCCCCGCTCTATCGATCGTCCTACCTCTTGCAGCCGATACTG
TGTTGGTGCTTACACAAATCATTTATTTTGCTCGAAGCTTCCCAAGGGGCTTGTCCATAGAAATTATTGGTGG
AACACGGACCTACATAGTGCCCGGGTTCTGACTTCAGCATGAAATTAGAGGTTGATTCAGAATCCGAGGCTTC
TCGCTGTGCAATCACTTACACAGACTGGTGTTTCTATTCCCAGCCAACCTAGGAGGGGCAGTGTTGAGCGCAA
AGATTGGTCCGGTAACAGCCGACAAATGCGACGCTTCTAACCTGGAGCTCGACAAGCATCCTCAATTGAAGCA
ATAGTCCACCTTAGTCAGAAACCACCGAATTACG"

,"ACTTCTCGCCCTCCAATCAATGCAACTTTCAAAGATATAGTCCCGGGGAGGTGTCACCTTGAACATGGTAA
GTTATGTCGTGGCTAAATTGTATAAGGTTTCCGGGCGTATTAAGGCCGTGCTCACTTCTGACGAGAACATGTG
TATTGGGGTTCTTCAACAAACAATCTTTGAGTTGGGGCGAGGTCGGTCCGAAGGGTCTAGTCCGAGAAGTAGT
GGGATGAGGAACGGACTCACGATTTTAGTGCCCCAGGAATCTGGTCAACGTATATAAGGTTATTCAGATCCAG
TGTGCTCCTCGCCCGGTTGCTACCATCCTCACAGACACTGTGCGTTTTTACTCCACGCACCATGGTAGAGGCA
TGTTTTGGGCAAAATCTATCGTCGGTAACAGCCCGTAACACTCTGCGTGTACATAACTTCAGGCTGCAACTCC
CCAAGGTGGCAATCCATATGTCCCCGTAGCCAGGTAACACCCGACATCTCA"
```

,"GTTCTACGCACTCAAACAGCTAACGTTTTTCAAATGAATTTCCGCCGGGGCGTGCACGACAAAGACGGTTA
AAGGCTTGTTCGGCTGAAAAATCGATAGTGTCCAGTGCGTTATTGCAAAGCGGGCCACCTCGCTTTGACGAAA
CGATGTTGTCTCGGCTCAAATCAGAAATCATTCGTATTGGCGGCGAGCGTTCGCACGGGGCGTTATCCAGTAG
AACTAGTTAGTAGGAGAACGGACACGGTATTGCCCAGCTGACGAATTTCACAACCGAACTTGAGAGTCTACTT
CGGACATCCAATGTCTGCAAGGCGGCGTTACCATGTTAGGCAACTGGTCGGGTTTCTAGTCCCACGGGCCATG
TAGCTAGTTTTGGGCAATACGATCTGGCGCCCAGCCCGACAACATCGCGGGTCATAAATCGTGGCCTGCAGCT
CCATGCGAAATCATTGTCCCTACGTAGGGTAAAACACGCGACGTAC"

,"CTGTCGCCCCTCATTCGACGGCAGTGTTCGATATTATACCGCCGGGGAGTGCCCGCAAGAACAGGAGGTAA
AGCTTCTGGGATAAAAATGACTAGTTTCCAGGGCTTGAAGGCGCGTCCCACCCTGACGGACAACAGTTGTTGG
AGCTCATCACTGAAACATGGATTGTGGGCAGGGCTTCCAAGTGCCGTTTCCGATAGAATCAATGTCGGCAACA
GGACCGGGATTATGCCCAGGTATCTGATTCAGGCAGATTGCGATGTATTTCTGAAGATAGGTTCTGCCGCTTG
TCTACTATTACCGTATTGGTGCGGCTCTTTACCCACGCACCATCGGAGGATGCTATGTTTTGGAGGCAAAGCT
TCGATGCGCTGAAGCCTGACATTCGGGGTCAAAAAATCCGAGGCACTGACCACGTCAATGTCAGAACTACTGT
CACCTCGTAGGTGCCCACGCACATTTCA"

,"ACTCGTGGCCCTCAACTAACCAACATTTATACTATATATCTCCCCGGGAGGTTCGACCGAGAACACGCGAT
AAAGGCCTGTACGTCTGAGAATCGATAGTTGTCCGGGCTTAGTAAGCGGGTCCTAGCCTTGACGGTAACTAGT
TTTGAGTAATCACATAAACTATATGAATTTGGCAGGCGCGTTCCTCGAAGGCGATTCCGATAAAATCGTTGGT
AGAGGAAGCACAGCTATTTGCCGAGAACTTGAATAGCAGGAATGAGGTTACTCGAGCAAGTCGTTCTCGCGCC
GTTCGTACATCTGTCACATGATCGGTCTCTATCCACAGACCTACTGCAGAGATGCAAGTTTTAACAAACGATT
CCTGCGACATCCGTAACTACGGAGTGCAATAATGCCAAGGCCAAGACGCTCCCAATAGTGACCGAGACGTGTC
CCATAAGTCGGATAGACAGACCCGACATTCA"};

    private static String s10490[] =

{"AGCGATGGGCCCGACTTTGTACGAAGTGTAACGACCGAAACGGCAAACGATGCAATTGGCGTAGGGAACCT
GCCAGTAATATTATTGGGTACAACAAAGTCGTTGCTTCGTAGCAGGTTGACACCTTAGTCACGGCGATGAGTA
TATACGACGGTGCCAGGTACAAGACGTTCATTACAATAGCGGCGCCCCAATACCCTCGATTTGGTGCACTATC
AAGATTAACATTTAATGCTATTTCGCAGTCAGGGTTCCGGCATTGTCTATACCATTGCCTGAAGTGTGCAAGT
ACGCGTCTGAGCCCCGAGACATAATCTGACGCGGGCTTTAGAACCTGAGCTGTCAGGATTGCCTATTCTAGGG
TGTTAGTCTCTTGCTATCAACAATATCATGGTATCCGTATCAATTCTCGACTATTTTTGCGGAGGATTTCACG
TGAACCCAGCAGTCCTCAGTAGTTCCTCGACAAAAGCTTGGGTCCAATGATACCCTAGAGGTTAGTG"

,"ACAGGCGGCCTGATGATGGAGCAGGAGCCATGAGACGCAGAAAGATTAGTATTGCGCAGTAACGGCATACG
TTCAAAGCAAAAATTTTCGCGTGTCGACAGACCCGCTTTCGTCAATCATGTGACCCTCAGTCAAGCGATGAGA
TAAGCGGTTCGATGACTGACTGTTTCAATTCCAAATTGTCACCCCAACACCCTCGTATGCCTTCTATCGAAAT
ACAATTAATGAATTGCGACACGTCGTCGCGAATAGTGCAATCCATAGTCACGGTGCAGGCTATGCGTCCTAGG
CCGAGGAATGAACTTTGCCGGCCTTGACTGTGCTGTAAGCGTCTGGTTGAGTTGTGATGCTCTTTGTATTCAT
CACAAATCGTGTACAAGATTTTACTGCTGCCAATTATTTTAGCGAGAGATTAGGACAGCGGAATCTCACTGTC
TTCAAAAAAGCAGCGTCAATGACAGCGACGGGTGAT"

,"AAGGGGCGCGATCTTAAGTGATCGTACGACGACCTATGAAACCGGAAAAGATTACAGCCTAGGATCTTGAG
CGATATTATTTTTGTCAAACAAGGACTGCTGTGTTCTAATCGAGTGCACTGACCAACATGGAGTATTAACCGG
TTAGCTTTTAGAAACATTTTTTTAATCAAATGAGTCCCCCCCAAACCTCGATATGCCTTTCATAATCACAATT
ATTTTTCGCATCACGGTTCCGCATAGTCAACCCCATTGCTAACGGTGTGCCAACTAATGGCCTTCTAGGCCGC
GCAGAGATAATTTTGCCGGCCCCGAGACTGCCCTGTAGAGCGTACGGCTCTTGGTTTGAACTCCTTGATAATC
ATCACAAAACTGGTTTAACTAGTCTATTGGACCTGCGATTTATTGTGCGCGGATTCTACCCCAGCAGCGAGTC
GCTCTTCTTGAAAACGCTTGTGGCTAAGTTACCCATAAGGTGAT"

,"AACGTGGGGTGCTTTAGACGTTGAGCGACACTGCAGCCGAAAAAGTAGCTAGGACGATGGCAGACTTCAGC
AAATTACTTTTGGGTTACTACAAGACGTGTCCTGCTTAATCGATGAACCTTAGTTCCAGCGTAAGATTACTCG
GTTGCAGCACAGAAATCTGTTCTTACAAAGTAGTCGTCCCCTAACCTCGACTTGGTCATCCGTGATACAAATT
AATGCGGCTTCCGGATTCTGTCTCGGACAATTGATCGAGCCTATGCTATGTACACGCAATCGCGCTCTAGGCG
CCGAGAACATATTTTGCCGGCCCTTTGGCGTAAGCTCCGACGATTGCGATATGTCGACGGTTTGATCTGCGGT

```
ATACTAATCAACAACCTGGTTCAAAGACATGCATTTTCGTATTATTTGCGGGATTCACTACTACACAGGAGCC
TCCACTATCTTCGAAAACTGATGGTGCAAGGTATACCACGAGGTGTTT"

,"AACGGGGCGTCTACATGATGCCGGAACCTGGAACGCGAAAAGATTACGCTATCGCATAGGCACGTCTTAGC
AGTATATATTTTGTGTCACAACAATCATGGCTCTACATGCAAGCTTCAAATTAGAATCAATCTTGTTATACAA
GCGGTTCGGAATCGAAAACCTGCTTCATACAAAATGCGCCCCCCAAAACTCCTGCATTGTCTTCCAGATGGAA
CTTAAATCGGTTTCGGCATGCAGATTCCCTGGGAATGACGAACCATTGTTACCGGGTCAAGGCCTGCGCCTCA
TAGCGCAGGAAATATTGAGCCGCTTGAATGATCACTGATGAGCTACTTGCTGCTTTTGAGGTTTGATCCTCCG
GTAATTATACAACACTGGTTTATCAGATTAATGTCGCATTATTTTAGCGGGTTGCAGGACCATCACGGAGTGA
CTGTATCTGCTCGAAAGACCTGATGGTACAGTTTAACCACGGAGTGT"};

    private static String s10500[] =

{"CCAAGATTCCCGGTAATCAAATCACTAAGACTTTTAGCCACCCGCTCGCCAATCCCTAGGACACGATATGT
TTGCTGTCCCCTCAGCTTAGCACGACGTATTCATAATCTCCCCTTCTATCCGCATCAGCGATTCATCCACTTC
GTCAAAAGCTTAGTGAGTGCGAAGTCATCACCTCCCTAAACTCACCGTCCGGTTCATGATTTCCCCGTCTCAG
GACTATTCAAGACCGATGTTTAATATACCAACTTGCTCCTTTGTTCTCACAGTGACTTGCCCGCGCACCAATC
AACCCCCTGCATCCTCTATATAAGAGGGTAGCCATGCAAGGTTCGTGGAGAAGATGTGCAGATATTCGCGCGA
ACAGGTGCTTCGACTCTTGCTAACCCTTATATCTCAGGGTCGATGCATTGGTTAGCCCTTGGAGCCCACACGA
GACCGGGACGCACCCAGCCCCCCTCTGAATGGGCGTAGCTTCCTAGCTGGTTATAGGAGTAACGTCGGCGGGT
T"

,"CCAAGACGTGATTTACGGTATTCAAGATCCCATAAGCTTTAGCCCCACTCCCACTCTGGCAAGTTCGTGTT
TGTTCCACGTCATATTAGCGTGCACTGGCTTTAATTTTCTCTTCGTGCGTTTTACGCATAGCTTCCCCTGACC
GCAAGCTAATGATCGAATCGAAATCCCGTCCCAAAACCCCACGTGCGTCAAATCTATCCTCCTCATCATGGAA
ATTGCTATGTCGATATGTAAATATAACACTTCGCGTCCATGATTCCGATCAGGATCGGCGCCGAGCGCACATC
CCCCTCTCTCCTTTAAGAGAGAGTGAGCACACGACAAGGCTTGGCCCTAACGCAGTGTCCACAAATTTGCCGC
GCAACCGGGGACAAGCTGAGTCGTCCTACTGTCTCAAGATAGCAGTCTTGGTTGTCCTCGAGCCCACCACGAG
ACCAGCCGTCCCCAGGCTCCCGTCCTCGGATTGTGTGGATTCTCATTCCATTAGGAATGAATGTTGGTGGGAT
T"

,"CCAAGGGTGTTTTACAGTACTAAATACATAAATATTTAAGCCCACCGTCCCCATCCCGAGACAGTGTGTTT
GCTTCGTTCATTTACCTGCGCTTCATTCTCCTCCTGTAGCTAGAACGACTGTATCGTACACTTGCATGATGAA
GTAGTGCCGAAGTGAAGTCCATCCCAAACATCCCTCCCGTCAATGATTTTCCCTCCTGAGCATTCAATACAAA
GTGTTAACTAACAAACTTGCGTTTCTTTGTTCCTATACAAGTGCGCCGCGGAGCATAACCCCCCTTACGCTTA
TTAACAGAGTGATCACTGCAAGAGGTCTGTCCGCTTGCACGTGCCACAACTAATTCTGCGTCTCAACCGGGAA
GCATAGCACTCTTTCTCTATCTTAATTGGGAGTGCTTGTTCCTGGGCTCCCCAGAGACCAGACGCCCCGACCC
TCACGCGCGAGGTGGCGTGATTGCCTTATGTCATTTTTAGTCATGCATGTGCGGTGT"

,"ACCGTATGTAGATATTAGAAACACATAATGCTTTACTCCTCCCGACGCCCATCCCGGAGATAGTGATGCTT
CCAGCTCATAACACGCATTGCAATCTAAATATCCTCCTTTGCTCTATCGACTTTGTATCTACACCTCCGCGCA
ATGAACTAGATGGACCGGAGTGACTCTCTCCCAAAGACTCTCGGTCGCCTAGTGATTAGCTCCCCTTCGGTTC
CAAGAACGATTGTATAATACCATCTTGTCCTTTGTCCTAAACCTGCTGGCACCGGGGATAACCCCTCCTTCCT
CTATTAAAAGTAGGGTCAGCAAAGGTATTGAGCCGAGCCATGTGCCCCCGAAATTTTCGCGCCGCAACCGGAG
ATGGACATCTACTTGTCAAATCATTTTAATAGCATTGACTTCTGTGAGCACTGAGGGCGAACCAGGACGAGCC
GCCCACCCTCGCTCGAATTGCGTGGTATCCTTATCGATATAGCAGCATTGGGGGTT"

,"CGAGAAGTTTAGGTTCAAATCTATAAATGCTTATGCCCCCGACTCACATGCCCAGGAAGAATTCATTGCTT
CCAATGTCACTTAGCAACCGATGTCTTATATTTTCCCTTCCCCTCAGCTCGACTCGCTGTTCTTAACCATGCC
AAAGAGTGTGATAGCTGATCAACCCACTCCCCAAAACTCACCGCTCTGCAATTGTATTCTCCTCGCCCAGTAT
TCATAGATCGAATGTATTTTCACACTTCTGTTTGTCCAAAAGAGATCGGGCCGCTGGATGTAAAAACCCCCCT
TTTTATTTAAAAAAGGGTCCAATGAAAAGTCTGTCGCCACAAGCCTGGCTCACAAATATTTTGCGCGACCGGG
ACGATTGCCATCCACAAACTATACTTACCTACAACTGTTTTGGTTGCCCTTGGAGCCCCACTATGACACGACG
CCCCTAGCCCCCCCCCCTCGAGTTGTTGATTCTTATTTCATCTAGCATAACGTTGTGGGTT"};

    public static String[] get_sequence(int n, int l){
```

```
String a[] = new String[n];
if (l == 10){
    for (int i = 0; i < n; i++){
        a[i] = s1010[i];
    }
}
else if (l == 20){
    for (int i = 0; i < n; i++){
        a[i] = s1020[i];
    }
}
else if (l == 30){
    for (int i = 0; i < n; i++){
        a[i] = s1030[i];
    }
}
else if (l == 40){
    for (int i = 0; i < n; i++){
        a[i] = s1040[i];
    }
}
else if (l == 50){
    for (int i = 0; i < n; i++){
        a[i] = s1050[i];
    }
}
else if (l == 60){
    for (int i = 0; i < n; i++){
        a[i] = s1060[i];
    }
}
else if (l == 70){
    for (int i = 0; i < n; i++){
        a[i] = s1070[i];
    }
}
else if (l == 80){
    for (int i = 0; i < n; i++){
        a[i] = s1080[i];
    }
}
else if (l == 90){
    for (int i = 0; i < n; i++){
        a[i] = s1090[i];
    }
}
else if (l == 100){
    for (int i = 0; i < n; i++){
        a[i] = s10100[i];
    }
}
else if (l == 110){
    for (int i = 0; i < n; i++){
        a[i] = s10110[i];
    }
}
```

```
else if (l == 120){
    for (int i = 0; i < n; i++){
        a[i] = s10120[i];
    }
}
else if (l == 130){
    for (int i = 0; i < n; i++){
        a[i] = s10130[i];
    }
}
else if (l == 140){
    for (int i = 0; i < n; i++){
        a[i] = s10140[i];
    }
}
else if (l == 150){
    for (int i = 0; i < n; i++){
        a[i] = s10150[i];
    }
}
else if (l == 160){
    for (int i = 0; i < n; i++){
        a[i] = s10160[i];
    }
}
else if (l == 170){
    for (int i = 0; i < n; i++){
        a[i] = s10170[i];
    }
}
else if (l == 180){
    for (int i = 0; i < n; i++){
        a[i] = s10180[i];
    }
}
else if (l == 190){
    for (int i = 0; i < n; i++){
        a[i] = s10190[i];
    }
}
else if (l == 200){
    for (int i = 0; i < n; i++){
        a[i] = s10200[i];
    }
}
else if (l == 210){
    for (int i = 0; i < n; i++){
        a[i] = s10210[i];
    }
}
else if (l == 220){
    for (int i = 0; i < n; i++){
        a[i] = s10220[i];
    }
}
else if (l == 230){
```

```
    for (int i = 0; i < n; i++){
        a[i] = s10230[i];
    }
}
else if (l == 240){
    for (int i = 0; i < n; i++){
        a[i] = s10240[i];
    }
}
else if (l == 250){
    for (int i = 0; i < n; i++){
        a[i] = s10250[i];
    }
}
else if (l == 260){
    for (int i = 0; i < n; i++){
        a[i] = s10260[i];
    }
}
else if (l == 270){
    for (int i = 0; i < n; i++){
        a[i] = s10270[i];
    }
}
else if (l == 280){
    for (int i = 0; i < n; i++){
        a[i] = s10280[i];
    }
}
else if (l == 290){
    for (int i = 0; i < n; i++){
        a[i] = s10290[i];
    }
}
else if (l == 300){
    for (int i = 0; i < n; i++){
        a[i] = s10300[i];
    }
}
else if (l == 310){
    for (int i = 0; i < n; i++){
        a[i] = s10310[i];
    }
}
else if (l == 320){
    for (int i = 0; i < n; i++){
        a[i] = s10320[i];
    }
}
else if (l == 330){
    for (int i = 0; i < n; i++){
        a[i] = s10330[i];
    }
}
else if (l == 340){
    for (int i = 0; i < n; i++){
```

```
            a[i] = s10340[i];
        }
    }
    else if (l == 350){
        for (int i = 0; i < n; i++){
            a[i] = s10350[i];
        }
    }
    else if (l == 360){
        for (int i = 0; i < n; i++){
            a[i] = s10360[i];
        }
    }
    else if (l == 370){
        for (int i = 0; i < n; i++){
            a[i] = s10370[i];
        }
    }
    else if (l == 380){
        for (int i = 0; i < n; i++){
            a[i] = s10380[i];
        }
    }
    else if (l == 390){
        for (int i = 0; i < n; i++){
            a[i] = s10390[i];
        }
    }
    else if (l == 400){
        for (int i = 0; i < n; i++){
            a[i] = s10400[i];
        }
    }
    else if (l == 410){
        for (int i = 0; i < n; i++){
            a[i] = s10410[i];
        }
    }
    else if (l == 420){
        for (int i = 0; i < n; i++){
            a[i] = s10420[i];
        }
    }
    else if (l == 430){
        for (int i = 0; i < n; i++){
            a[i] = s10430[i];
        }
    }
    else if (l == 440){
        for (int i = 0; i < n; i++){
            a[i] = s10440[i];
        }
    }
    else if (l == 450){
        for (int i = 0; i < n; i++){
            a[i] = s10450[i];
```

```
            }
        }
        else if (l == 460){
            for (int i = 0; i < n; i++){
                a[i] = s10460[i];
            }
        }
        else if (l == 470){
            for (int i = 0; i < n; i++){
                a[i] = s10470[i];
            }
        }
        else if (l == 480){
            for (int i = 0; i < n; i++){
                a[i] = s10480[i];
            }
        }
        else if (l == 490){
            for (int i = 0; i < n; i++){
                a[i] = s10490[i];
            }
        }
        else if (l == 500){
            for (int i = 0; i < n; i++){
                a[i] = s10500[i];
            }
        }
        return a;
    }
}
```

BIOGRAFI PENULIS

Penulis memiliki nama lengkap Muhammad Luthfi Shahab. Penulis lahir di Malang, pada tanggal 31 Maret 1995. Penulis telah menempuh pendidikan di SD Negeri Gondanglegi Wetan 1 (2001-2007), MTs Negeri Malang 3 (2007-2009), dan MA Negeri 3 Malang (2009-2011).

Setelah lulus MA, penulis mendaftar di Jurusan Matematika ITS melalui jalur SNMPTN undangan dan tercatat sebagai mahasiswa Matematika ITS dengan NRP 1211100047. Selama menempuh kuliah di Jurusan Matematika ITS, penulis pernah menjadi asisten dosen serta aktif di organisasi kemahasiswaan. Penulis pernah aktif di organisasi HIMATIKA-ITS sebagai staff DAGRI periode 2012-2013 dan staff SAINSTEK periode 2013-2014. Penulis juga pernah menjadi anggota Steering Committee Padamu Himatika periode 2013-2014. Setelah lulus pada September 2015, penulis langsung melanjutkan jenjang Magister di Jurusan yang sama dan tercatat dengan NRP 1215201010.

Segala saran dan kritik yang membangun selalu penulis harapkan untuk kebaikan ke depannya. Penulis dapat dihubungi melalui nomor +6287751132372 atau melalui email shahab. luthfi@gmail.com.